Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Sóc de l'EETAC: desenvolupament d'una aplicació per a iPhone i Android

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTORS: Joan Marc Amadó Medina
Andreu Martínez Alcón

DIRECTOR: Rafael Vidal Ferré
SEGON DIRECTOR: Toni Oller Arcas

DATA: 4 de desembre de 2012

Títol: Sóc de l'EETAC: desenvolupament d'una aplicació per a iPhone i Android

Autors: Joan Marc Amadó Medina
Andreu Martínez Alcón

Director: Rafael Vidal Ferré
Segon Director: Toni Oller Arcas

Data: 4 de desembre de 2012

Resum

Cada dia és més habitual veure la gent utilitzant telèfons intel·ligents per accedir a diferents serveis, desplaçant l'ús dels ordinadors en favor dels telèfons mòbils. En aquest escenari, l'aplicació desenvolupada en aquest TFC, "Sóc de l'EETAC", permet donar un determinat conjunt de serveis i mecanismes per mantenir un vincle entre els estudiants, els antics alumnes i la seva Escola, l'Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels, com ara el servei de missatgeria i localització. Aquests serveis s'entrellacen amb les xarxes socials, especialment amb LinkedIn i Twitter.

A part de mantenir el contacte, un dels objectius d'aquest TFC és proporcionar una eina per donar a conèixer l'Escola a la gent que pugui estar interessada en estudiar-hi, per exemple oferint informació estadística sobre el nombres de titulats o els lloc de treball que ocupen.

L'aplicació disposa dels canals d'informació habituals de l'Escola, tant els canals de les seves pàgines web, com els canals a través de les xarxes social. També disposa d'un conjunt de dades informatives sobre els alumnes, educació i món laboral de l'Escola. Tampoc hi falten els principals serveis interns de les intranets de l'Escola. A part dels ja mencionats serveis de missatgeria i localització per a estudiants i titulats.

L'aplicació "Sóc de l'EETAC" ha estat desenvolupada per al dispositiu mòbil iPhone i per als telèfons amb Android.

Title: I'm from EETAC: development of a mobile application for iPhone and Android

Authors: Joan Marc Amadó Medina
Andreu Martínez Alcón

Director: Rafael Vidal Ferré
Second Director: Toni Oller Arcas

Date: December 4, 2012

Overview

Each day is more common to see people using smart phones in order to access to different services, displacing the use of computers in favour of mobile phones. The application "I'm from EETAC" allows give to the students and ex-students of the Castelldefels School of Telecommunications and Aerospace Engineering, a certain pack of services. These services are intertwined with social networks, especially with LinkedIn and Twitter.

In addition of keeping the contact, one of the objectives of this TFC is to give a tool in order to introduce the School to people who can be interested in studying on it, for example, giving statistical information about the number of graduates and their actual jobs.

The application has the usual information channels of the school : the channels of their web pages and the channels through social networks. It also has a set of informative data about students, education and the school labour world. Are also included the main internal intranet services of the school. As well as the messenger and locating students and graduates system mentioned before.

The Application "I'm from EETAC" has been developed for iPhone mobile device and for Android phones.

ÍNDEX

INTRODUCCIÓ 9

CAPÍTOL 1. VISIÓ GENERAL 11

1.1. Objectius 11

1.2. Tasques 11
1.2.1. Coneixements previs 12
1.2.2. Disseny de l'aplicació 12
1.2.3. Disseny dels protocols de comunicació 12
1.2.4. Desenvolupament de l'aplicació 13
1.2.5. Proves i correccions 13

CAPÍTOL 2. CAS D'ÚS 14

2.1. Escenari 14

2.2. Accés i Menú 14

2.3. Apartat d'Opcions 14

2.4. Apartat de l'Escola 15

2.5. Apartat Social 15

2.6. Apartat de Companys 15

2.7. Apartat de Missatges 16

2.8. Apartat de Dades 16

CAPÍTOL 3. ARQUITECTURA 17

3.1. Back-end 17
3.1.1. Servidor "Sóc de l'EETAC" (mòdul A) 17
3.1.2. Backoffice (mòdul B) 18
3.1.3. Bus SOA (mòdul C) 18
3.1.4. Bases de Dades (mòdul D) 19

3.2. Front-end 19
3.2.1. Web "Sóc de l'EETAC" (mòdul E) 19
3.2.2. Aplicacions mòbils "Sóc de l'EETAC" (mòdul D) 20

3.3. Xarxes socials i professionals 20
3.3.1. LinkedIn 20
3.3.2. Twitter 20
3.3.3. Facebook 21
3.3.4. Google+ 21
3.3.5. YouTube 21

CAPÍTOL 4. IMPLEMENTACIÓ DEL SERVIDOR 22

4.1. Llenguatge d'intercanvi d'informació estructurada 22
4.1.1. Escenari 22
4.1.2. XML 22
4.1.3. JSON 23
4.1.4. Elecció del llenguatge de comunicació estructurada 24
4.1.5. Implementació JSON al Servidor 24
4.1.6. Adreça web d'accés a la informació 25

4.2. Objectes 26
4.2.1. EetacPerson 26
4.2.2. ImHere 28
4.2.3. Message 29
4.2.4. WebsInput 29
4.2.5. StatisticsInput 30

4.3. Funcions 30
4.3.1. Funcions amb el mètode GET 30
4.3.2. Funcions amb el mètode POST 31
4.3.3. Funcions de testeig i control 33

4.4. Bases de dades temporals 34
4.4.1. Implementació 34
4.4.2. Bases de Dades 35

CAPÍTOL 5. IMPLEMENTACIÓ DE L'APLICACIÓ 36

5.1. Disseny i elecció dels mòduls de l'aplicació 36

5.2. Llibreries Externes 37
5.2.1. Android 37
5.2.2. iOS 45

5.3. Llibreries Internes 46
5.3.1. Android 46
5.3.2. iOS 47

5.4. Implementació i creació de vistes 55
5.4.1. Android 55
5.4.2. iOS 59

5.5. Implementació del mòdul d'Accés i Menú 63
5.5.1. Funcionament 64
5.5.2. Android 64
5.5.3. iOS 66

5.6. Implementació del mòdul L'Escola 68
5.6.1. Funcionament 68
5.6.2. Android 69
5.6.3. iOS 70

5.7. Implementació del mòdul Missatges 72
5.7.1. Funcionament 72
5.7.2. Android 74
5.7.3. iOS 77

5.8. Implementació del mòdul Companys 80
5.8.1. Funcionament 80

5.8.2. Implementació Android ... 81
5.8.3. Implementació iOS ... 84

5.9. Implementació del mòdul Social ... 86
5.9.1. Funcionament ... 86
5.9.2. Android ... 87
5.9.3. iOS ... 89

5.10. Implementació del mòdul Dades ... 90
5.10.1. Funcionament ... 91
5.10.2. Android ... 91
5.10.3. iOS ... 92

5.11. Implementació del mòdul Opcions ... 93
5.11.1. Funcionament ... 93
5.11.2. Android ... 94
5.11.3. iOS ... 97

CAPÍTOL 6. PLANIFICACIÓ I DESENVOLUPAMENT ... 101

6.1. Disseny ... 101

6.2. Assignació de tasques ... 101

6.3. Desenvolupament ... 102

6.4. Presentació ... 102

CAPÍTOL 7. CONCLUSIONS ... 103

7.1. Resultats ... 103

7.2. Millores futures ... 104
7.2.1. Accés a l'aplicació ... 104
7.2.2. Menú ... 104
7.2.3. Mòdul L'Escola ... 104
7.2.4. Mòdul Missatges ... 105
7.2.5. Mòdul Companys ... 105
7.2.6. Mòdul Social ... 105
7.2.7. Mòdul Dades ... 105
7.2.8. Mòdul Opcions ... 106
7.2.9. Altres millores ... 106

7.3. Impacte mediambiental ... 107

CAPÍTOL 8. BIBLIOGRAFIA ... 108

8.1. Referències Android ... 108

8.2. Referències iOS ... 108

8.3. Referències Servidor ... 109

8.4. Referències Comunes ... 109

8.5. Xarxes socials i professionals de l'Escola ... 110

ANNEX I. INTERFÍCIE GRÀFICA 111

I.1. Índex de figures 111

I.2. Inici 112

I.3. Accés a l'aplicació 112

I.4. Menú 113

I.5. Mòdul L'Escola 114

I.6. Mòdul de Missatges 115

I.7. Mòdul Companys 119

I.8. Mòdul Social 120

I.9. Mòdul de Dades 123

I.10. Mòdul d'Opcions 124

ANNEX II. ANDROID 127

II.1. Estructura d'una aplicació 127
II.1.1. Android Manifest 127
II.1.2. Cicle de vida d'una aplicació 128
II.1.3. Activity 128
II.1.4. Intent 129
II.1.5. View 130
II.1.6. Estructura de carpetes 132

II.2. Maps 133
II.2.1. Passos per aconseguir l'API Key de Google 134
II.2.2. Objectes de Mapes 135
II.2.3. Localització 135

II.3. Instal·lació GreenDroid 136

II.4. Classe SendRecive 137

ANNEX III. ACRÒNIMS I DEFINICIONS 142

III.1. Acrònims 142

III.2. Definicions 143

INTRODUCCIÓ

Avui dia, l'ús que li donem als telèfons mòbils no s'acaba en realitzar trucades o enviar missatges de text, sinó que a mesura que la tecnologia ha anat avançant, aquests dispositius han anat evolucionant i amb ells la nostra forma d'utilitzar-los. La seva potència supera amb escreix la dels primers ordinadors personals i això ha permès l'expansió de serveis i l'aparició de noves funcionalitats: càmera de fotos i vídeo, accés a internet, geoposicionament [78], mapes, reproductors multimèdia, jocs, etc.

Per altra banda, l'Escola vol desenvolupar un sistema que permeti mantenir el contacte amb antics alumnes titulats/des de l'EETAC, abans denominada EPSC i originalment EUPBL, i obtenir informació laboral d'aquests alumnes a través del grup "EETAC Alumni" que l'Escola disposa a la xarxa social LinkedIn, per poder realitzar un estudi amb aquestes dades i aprofitar-les per promocionar l'Escola a nous potencials estudiants.

Per resoldre aquestes necessitats s'ha pensat en definir un sistema amb dues parts ben diferenciades. La primera part a desenvolupar consisteix en una eina de recollida d'informació procedent de diferents fonts de dades, entre elles les xarxes socials i bases de dades pròpies de l'Escola (Alumni i Mitra). La segona part és l'encarregada de desenvolupar una aplicació mòbil per comunicar els diferents usuaris d'aquest sistema utilitzant les dades que s'han validat i normalitzat sobre l'eina de recollida d'informació (primera part de la feina).

Aquest TFC es centra en la segona part del sistema i consisteix en el desenvolupament de dues aplicacions per a telèfons mòbils, una per Android i una altra per a iPhones, anomenada "Sóc de l'EETAC". Aquest sistemes operatius són els majoritaris en el mercat de telèfons intel·ligents, on representen el 89,9% [34].

Per mantenir el contacte, l'aplicació s'ha aprofitat de la xarxa professional de LinkedIn per fer un recull de dades dels titulats de l'EETAC que hi figuren, actualment hi ha 578 membres del grup EETAC Alumni [35], i geoposicionar-los tant a ells com els seus llocs de treball, utilitzant els mapes d'Apple o Google Maps, de manera que un usuari de l'aplicació pot localitzar a un altre company fàcilment. Un altre element important per mantenir aquest contacte és el servei de missatges del qual disposa l'aplicació, on cada usuari podrà comunicar-se amb altres companys.

L'aplicació també és una eina per promocionar i conèixer les diferents novetats que succeeixen a l'Escola connectant directament amb el canal de Twitter [36] i LinkedIn de l'EETAC, en els quals es pot participar activament; també es té accés als canals Facebook [37], Google+ [38] i YouTube [39]. També pretén mostrar un seguit de gràfiques i estadístiques per promocionar l'Escola, per exemple, donant informació de les diferents sortides professionals de totes les

titulacions que s'imparteixen a l'EETAC, amb aquesta informació es podrà veure quina és l'evolució dels estudiants un cop han acabat els seus estudis.

Per a que tot aquest sistema funcioni, s'ha desenvolupat parcialment un servidor que permet la comunicació entre els diferents dispositius mòbils i s'ha dissenyat un protocol per dur a terme aquesta comunicació.

Al capítol 1, es presenta l'escenari amb una solució a la demanda de l'EETAC per millorar la comunicació amb els seus estudiants i antics alumnes. Aquesta demanda propicia el desenvolupament d'aquest TFC. En aquest capítol també s'estudien els objectius a complir, així com els procediments i tasques a realitzar per aquest projecte.

Al capítol 2, es presenta com a cas d'ús, un exemple, on un usuari realitza una sèrie de consultes a l'aplicació. Aquest exemple, presentat a mode d'historieta amb l'Alícia i en Blas com a personatges, exposa un cas concret de l'ús de l'aplicació "Sóc de l'EETAC".

Al capítol 3, es presenta l'arquitectura global del sistema emprat per aquest projecte, donant a conèixer cadascun dels elements i mòduls que el formen.

Al capítol 4, s'exposa la implementació de la part del servidor relacionada amb les aplicacions mòbils. Es presenten els objectes, funcions i protocols utilitzats per la comunicació entre l'aplicació i el servidor "Sóc de l'EETAC".

Al capítol 5, s'exposa la implementació de l'Aplicació "Sóc de l'EETAC" amb tot el seu contingut i funcionament, tant de l'aplicació desenvolupada per a iPhone com l'aplicació desenvolupada per a telèfons amb Android.

Al capítol 6, s'explica com s'ha planificat aquest projecte i es mostraran les fases de disseny i desenvolupament del projecte.

Al capítol 7, es fa una valoració del projecte realitzat i s'analitzen diferents aspectes desenvolupar per millorar l'aplicació "Sóc de l'EETAC" en un futur.

Per últim al capítol 8, es troba la bibliografia i referències emprades per la realització d'aquest projecte.

CAPÍTOL 1. VISIÓ GENERAL

El projecte “Sóc de l’EETAC” que consta de dos mòduls a desenvolupar:

- Una aplicació per a iPhone i Android [66].
- Una eina de recollida d’informació a les xarxes professionals de titulats/des.

En aquest projecte ens centrarem en el primer mòdul. El segon mòdul es realitzarà en un TFC diferent i desenvolupa el servidor. Tot i que al primer mòdul es realitzarà una part del servidor per al funcionament de les aplicacions mòbils.

1.1. Objectius

Es vol crear una aplicació per a Android i una altra per a iPhone que permeti la comunicació entre els usuaris d’aquestes i doni un determinat nombre de serveis a antics alumnes titulats de l’EETAC.

L’aplicació haurà de tenir les següents funcions:

- Facilitar la recollida d’informació laboral dels antics alumnes de l’EETAC a través de la xarxa social de LinkedIn.
- Facilitar la comunicació dels antics alumnes a través d’un servei de missatgeria.
- Geoposicionar [78] els usuaris de l’aplicació i els seus llocs de treball.
- Visualitzar els continguts de les xarxes socials que utilitza l’Escola: LinkedIn, Twitter i interactuar amb elles.
- Visualitzar tot un seguit de dades i estadístiques referents a l’Escola per promocionar-la.
- Connectar-se i intercanviar dades amb el servidor del sistema “Sóc de l’EETAC”.

També s’haurà de dissenyar un protocol de comunicació entre les diferents aplicacions mòbils i el servidor.

1.2. Tasques

En aquest apartat veurem quines són els requisits previs i les tasques a realitzar per desenvolupar l’aplicació “Sóc de l’EETAC”.

1.2.1. Coneixements previs

1.2.1.1. iPhone

Les aplicacions per iPhone utilitzen el sistema operatiu iOS desenvolupat per Apple. Aquest sistema utilitza el llenguatge de programació Objective-C per desenvolupar aplicacions.

Serà necessari aprendre aquest llenguatge de programació ja que no es tenen coneixements previs sobre el mateix, per assolir aquesta tasca he llegit diversos llibres [12] [13] [14]. Per programar les aplicacions s'utilitza l'eina Xcode [103], caldrà familiaritzar-se amb aquest entorn.

1.1.1.1. Android

Les aplicacions per telèfons Android, com el seu nom indica, utilitzen el sistema operatiu Android desenvolupat per Google, aquest sistema utilitza el llenguatge de programació Java i per desenvolupar les seves aplicacions fa us de l'eina Eclipse [77].

Tot i que durant al llarg de la carrera s'ha treballat en alguna ocasió amb aquest llenguatge ha calgut reforçar i ampliar coneixements del mateix, així com aprendre les noves funcionalitats que ha agregat Google per poder desenvolupar aplicacions pel seu sistema operatiu.

1.2.2. Disseny de l'aplicació

S'haurà de realitzar un disseny de l'estructura de l'aplicació per aconseguir els objectius del projecte i s'haurà de seguir unes mateixes pautes a l'hora de desenvolupar les vistes dels diferents apartats dels quals disposarà l'aplicació, perquè hi hagi una semblança amb l'aplicació desenvolupada per iOS (iPhone) i l'aplicació desenvolupada per a telèfons amb sistema operatiu Android.

1.2.3. Disseny dels protocols de comunicació

El sistema consta de varis mòduls i tots ells s'han de poder comunicar. S'haurà de crear un protocol perquè totes les parts puguin disposar de la informació necessària pel seu funcionament.

Per tant, s'haurà de crear i pactar un sistema de peticions i respostes per a cada mòdul, utilitzant el mètode més adequat en cada cas.

1.2.4. Desenvolupament de l'aplicació

S'hauran de realitzar les següents tasques per desenvolupar l'aplicació:

- Crear una interfície per mostrar la informació i serveis desitjats de l'EETAC.
- Construir un sistema de missatgeria.
- Crear una interfície per geoposicionar [78] usuaris i llocs de treball.
- Mostrar i integrar diferents xarxes socials.
- Mostrar dades significatives per promocionar l'Escola.
- Recollir dades dels usuaris per completar diverses informacions al sistema.

1.2.5. Proves i correccions

Un cop desenvolupada l'aplicació serà necessari realitzar diverses proves per comprovar el seu funcionament. També serà necessari realitzar proves del protocol de comunicació i comprovar el funcionament de tot el sistema sencer.

Per últim, s'hauran de corregir i ajustar problemes que hagin sorgit durant el desenvolupament de l'aplicació i/o el disseny dels protocols de comunicació.

CAPÍTOL 2. CAS D'ÚS

En aquest capítol s'exposarà un exemple de com un alumne titulat podria utilitzar l'aplicació "Sóc de l'EETAC" i com treure-li profit. Primer de tot es presentaran els personatges, i seguidament s'aniran descrivint els diferents apartats de l'aplicació tot seguint una historieta.

2.1. Escenari

En Blas acaba de finalitzar la carrera i es disposa a buscar treball. A la universitat li han recomanat l'aplicació "Sóc de l'EETAC" ja que amb aquesta eina podrà extreure informació d'altres titulats que ja han acabat els seus estudis i ja estan incorporats en el món laboral, d'aquesta manera podrà observar de forma senzilla i directe els possibles camins que pot escollir.

En Blas disposa d'un iPhone i vol provar l'aplicació, per fer-ho haurà de descarregar-se-la des de l'App Store [69], des del seu propi iPhone o des de l'ordinador personal a través del programa l'iTunes [87] i instal·lar-la.

2.2. Accés i Menú

En Blas, a l'obrir per primer cop l'aplicació, li demana que s'identifiqui mitjançant la seva compte de LinkedIn o entri com a convidat.

En Blas ja té compte de LinkedIn, amb el qual disposa d'un currículum en línia i pot mantenir el contacte amb professionals del sector, però sinó fos el cas, hauria de crear un perfil a la web de LinkedIn.

Un cop identificat a l'aplicació el primer que es troba és el menú principal que disposa dels següents apartats: L'Escola, Missatges, Companys, Social, Dades i Opcions.

2.3. Apartat d'Opcions

El primer que li agrada fer a en Blas es configurar-se les aplicacions al seu gust. Per a fer-ho entrarà a l'apartat d'Opcions.

Dins d'aquest apartat podrà veure la seva informació de l'EETAC (correu electrònic, titulació, data de l'obtenció del títol), podrà desconnectar-se de la xarxa de LinkedIn si no vol utilitzar més l'aplicació, podrà vincular la seva compte de Twitter si vol participar en el servei de missatgeria o repiular [99] la

informació que més li agradi i també podrà definir quina informació vol compartir amb la resta d'usuaris de l'aplicació.

2.4. Apartat de l'Escola

En Blas ha descobert que a l'apartat de l'Escola es pot trobar un llistat d'enllaços relacionats amb la Universitat, ha trobat interessant un parell d'apartats:

- **Portal de Titulats/des:** un espai on podrà trobar els seus companys d'estudis, contactar-los, mantenir la relació amb l'Escola i conèixer de primera mà els anuncis i les activitats especialment dirigides als titulats i titulades.
- **Màsters:** un recull de tots els màsters que es s'imparteixen a l'EETAC. Potser més endavant es decideix a cursar-ne algun.

2.5. Apartat Social

Ara en Blas decideix entrar a l'apartat Social on pot veure el "Timeline" del compte de Twitter de l'Escola, i com que ha decidit vincular l'aplicació amb el seu compte de Twitter ara pot repiular qualsevols de les piulades que li semblin interessants.

A l'apartat de LinkedIn en Blas estarà al dia de la informació que es publica al perfil de L'EETAC. També podrà consultar la informació que l'Escola publica a altres xarxes social com la de Facebook, Google+ i YouTube.

2.6. Apartat de Companys

En Blas ha vist un comentari al perfil de LinkedIn de l'Escola de l'Alícia (una antiga companya de classe), on buscava gent recent titulada per treballar a la mateixa empresa on és ella.

L'Alícia també disposa de l'aplicació "Sóc de l'EETAC" després d'haver entrat a la botiga en línia Google Play i haver-se descarregat i instal·lat l'aplicació. Un cop identificada a l'aplicació, l'Alícia ha compartit la localització des de l'apartat d'Opcions, tant d'ella com de la seva empresa.

En Blas ha vist l'oferta i creu tenir el perfil adequat per presentar-s'hi, així que decideix mirar a l'aplicació i entrar a l'apartat de Companys, on pot veure si l'empresa està molt lluny d'on ell es mou. L'oferta l'interessa ja que veu bona la ubicació de l'empresa, i decideix posar-se en contacte amb l'empresa a través de la xarxa de LinkedIn.

Malauradament no l'escullen per treballar a l'empresa de l'Alícia.

Aprofitant que ja coneix l'apartat de Companys i que també ha trobat l'Alícia, en Blas decideix aprofitar el moment per enviar-li un missatge. Per fer-ho només ha de d'accedir a la informació que comparteix l'Alícia, i prémer el botó d'enviar un missatge.

2.7. Apartat de Missatges

En Blas ara pot enviar un missatge a l'Alícia, i un cop el té escrit pot decidir si li envia a través del sistema de l'aplicació o a través de la xarxa de Twitter. En aquest apartat de Missatges també pot consultar els missatges rebuts tant de l'aplicació com de Twitter.

2.8. Apartat de Dades

En Blas s'havia quedat un pèl decebut per no haver obtingut la feina a l'empresa de l'Alícia i entra a l'apartat de Dades i veu una gràfica molt interessant: el 97% dels alumnes titulats a l'EETAC troben feina abans de 3 mesos d'haver-se titulat. Al veure això en Blas s'anima i segueix buscant ofertes de treball.

CAPÍTOL 3. ARQUITECTURA

En aquest capítol es presentarà l'esquema de tot el sistema del projecte "Sóc de l'EETAC" i es definirà els diferents mòduls i les seves principals funcions.

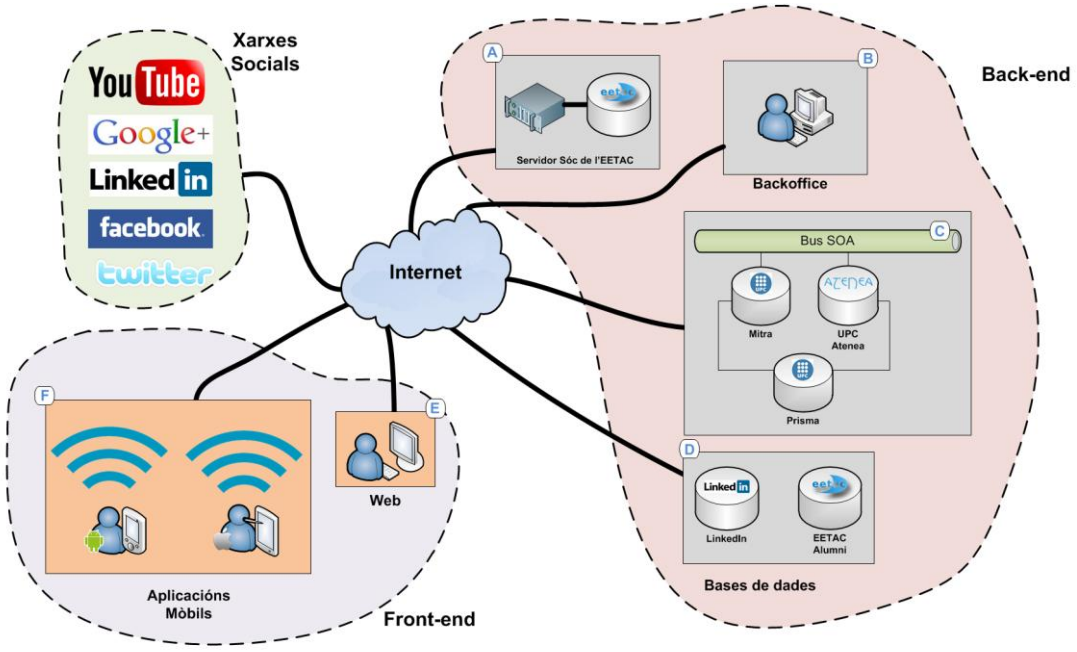

Fig. 3.1 Esquema de l'Arquitectura del projecte global "Sóc de l'EETAC"

L'arquitectura es dividirà en tres parts:

- Back-end
- Front-end
- Xarxes Socials.

3.1. Back-end

El back-end és la part del sistema que es cuida de realitzar totes aquelles operacions que no són perceptibles per l'usuari final i que són vitals pel funcionament general de tot el sistema.

3.1.1. Servidor "Sóc de l'EETAC" (mòdul A)

El servidor és l'encarregat de rebre i distribuir la informació procedent dels diferents mòduls. Tenim dues funcions ben diferenciades al servidor:

- Recollir dades dels mòduls Bus SOA (veure apartat 3.1.3) i Bases de Dades. Del primer mòdul bàsicament es recull la informació universitària de cada un dels alumnes de l'EETAC. I en el segon mòdul es recullen les dades públiques dels alumnes que utilitzin la xarxa professional de LinkedIn o estiguin registrats a la pàgina web de l'EETAC Alumni.
- Enviar i rebre dades dels mòduls Backoffice [71], Web, Aplicacions mòbils.

El Servidor disposa d'una base de dades persistent per emmagatzemar tota la informació recollida dels mòduls Bus SOA i Bases de Dades, i la informació procedent del mòdul Aplicacions mòbils.

El Servidor també disposa del servei web Apache Tomcat [67], que és l'encarregat de córrer el programa "Sóc de l'EETAC". Aquest programa està desenvolupat amb el llenguatge de programació Java a través de l'eina Eclipse [77]. El programa implementa un servei REST (REpresentational State Tranfer) d'intercanvi de peticions anomenat Jersey. Aquestes peticions són amb format de text JSON (JavaScript Object Notation) i concretament a través de l'estàndard Jackson [88] que ofereix el servei Jersey.

3.1.2. Backoffice (mòdul B)

La seva principal funció és administrar la base de dades del mòdul Servidor. Aquesta administració és realitza a través d'una pàgina web i en ella es pot realitzar varies operacions:

- Assignar coincidències entre les dades procedents del mòdul Bus SOA i la xarxa professional LinkedIn.
- Afegir, modificar o eliminar dades manualment de la base de dades del mòdul Servidor.

3.1.3. Bus SOA (mòdul C)

El Bus SOA a través del servidor Mitra és l'encarregat de proporcionar les dades universitàries de cada alumne de l'EETAC, aquest mòdul ens proporciona aquestes dades amb format de text CSV [74] (Comma-Separed Values).

```
Titulació;Estudiant;Quadrimestre d'obtenció;Data titulació;Quadrimestre d'ingrès;Correu electrònic
ENG TELEC 2NCICLE 01;XXXX YYYY, ZZZZ;2009-2;26/07/2010;2007-1;test@estudiant.upc.edu
ET AERO/ETT SIST 05;XXXX YYYY, ZZZZ;2010-1;26/01/2011;2007-1;test@estudiant.upc.edu
ET AERONÀUTICA 03;XXXX YYYY, ZZZZ;2009-2;21/06/2010;2006-1;test@estudiant.upc.edu
ETT SIST TELEC 00;XXXX YYYY, ZZZZ;2002-2;21/07/2003;2001-1;test@estudiant.upc.edu
ETT SIST TELEC 91;XXXX YYYY, ZZZZ;1995-1;29/02/1996;1992-1;test@alu-etsetb.upc.edu
ETT SIST TELEC 91;XXXX YYYY, ZZZZ;1994-2;07/09/1995;1991-1;
ETT SIST/ ET AERO 05;XXXX YYYY, ZZZZ;2007-1;07/02/2008;2005-1;test@estudiant.upc.edu
```

```
ETT TELEMÀTICA 00;XXXX YYYY, ZZZZ;2007-1;24/01/2008;2004-1;test@estudiant.upc.edu
MO AEROSPACE S&T;XXXX YYYY, ZZZZ;2009-1;01/06/2010;2008-1;test@estudiant.upc.edu
MU MASTEAM;XXXX YYYY, ZZZZ;2010-2;30/09/2011;2009-2;test@estudiant.upc.edu
```

Fig. 3.2 Exemple de format CSV [74] del mòdul Bus SOA

Els camps de dades rebuts al fitxer són els:

- La titulació obtinguda a l'EETAC.
- Els noms i cognoms de l'estudiant.
- El quadrimestre d'obtenció del títol.
- La data de l'obtenció del títol.
- El quadrimestre d'ingrés a l'EETAC.
- El correu electrònic de l'estudiant.

3.1.4. Bases de Dades (mòdul D)

En aquest mòdul es fa ús de l'API [68] (Application Programming Interface) de LinkedIn per accedir a la informació laboral pública de cada alumne, ja estigui inscrit al grup EETAC Alumni que l'Escola disposa a la xarxa professional LinkedIn, o a través de l'aplicació mòbil "Sóc de l'EETAC".

Aquesta informació laboral s'extreu a partir del servei REST [98] que ofereix l'API de LinkedIn en format JSON.

3.2. Front-end

El front-end és aquella part del sistema que es cuida d'interactuar amb l'usuari final, a través d'una interfície gràfica amigable, i realitzar totes aquelles operacions que l'usuari demanda, fent ús del back-end.

3.2.1. Web "Sóc de l'EETAC" (mòdul E)

Es tracta d'una pàgina web on es pretén promocionar l'Escola, oferint informació sobre els seus estudis i serveis. La finalitat d'aquesta web dintre el projecte és mostrar a l'usuari un estudi de les dades recollides a través de la xarxa professional de LinkedIn, per mostrar l'evolució dels seus titulats en el món laboral i ajudar als nous potencials estudiants a decidir-se a realitzar els seus estudis a l'EETAC.

3.2.2. Aplicacions mòbils "Sóc de l'EETAC" (mòdul D)

Aquest mòdul és el principal element d'interacció de l'usuari amb el sistema "Sóc de l'EETAC". És l'encarregat de recollir la informació laboral de la xarxa professional LinkedIn; de recollir i mostrar dades de posicionament dels usuaris de l'aplicació; de facilitar la comunicació entre usuaris de l'aplicació mitjançant un servei de missatges; de mostrar comunicats, esdeveniments i notícies d'interès a través de les diferents xarxes socials on està present l'Escola; de mostrar dades significatives de l'Escola i els alumnes que la componen.

3.3. Xarxes socials i professionals

Les xarxes socials juguen un paper molt important dins del projecte, en aquest apartat es definiran les xarxes on està present l'EETAC i quin paper desenvolupen dins el projecte.

3.3.1. LinkedIn

LinkedIn és una xarxa social orientada als professionals i els negocis. El principal propòsit d'aquesta xarxa és posar en contacte i establir col·laboracions entre professionals qualificats. LinkedIn permet la creació de perfils tant per a professionals com per a empreses i la creació de grups on apareixen les principals discussions de cada entitat.

LinkedIn juga el paper més important dins del projecte, aporta la informació laboral dels alumnes de l'Escola i permet la realització d'un estudi d'aquestes dades. També mostra les discussions de l'Escola i aporta els sistema d'autenticació al mòdul d'Aplicacions Mòbils.

3.3.2. Twitter

Twitter és una xarxa social que permet als seus usuaris enviar i llegir missatges de text d'una longitud màxima de 140 caràcters denominats piulades. El seu propòsit és promoure la comunicació i seguiment d'usuaris.

Dins del projecte, Twitter mostra les piulades de l'Escola permetent una interacció amb ells i s'integra al servei de missatgeria del mòdul Aplicacions Mòbils.

3.3.3. Facebook

Facebook és una xarxa social que permet afegir gent com a amics i enviar-los missatges, compartir enllaços, fotografies i vídeos, entre d'altres coses; tot a través del seu mur d'esdeveniments.

Aquesta xarxa dins del projecte té un paper secundari, i dins del mòdul Aplicacions Mòbils només té la funció de mostrar el mur del que disposa l'Escola a Facebook.

3.3.4. Google+

Google+ és una xarxa social que permet afegir gent i classificar-los en cercles de d'amistats. També es poden crear cercles d'altres qüestions, com ara, quedades, interessos, missatges, etc. Google+ també integra el servei de xat existent a GMail.

Aquesta xarxa dins del projecte té un paper secundari, i dins del mòdul Aplicacions Mòbils només té la funció de mostrar les publicacions de que disposa l'Escola a Google+.

3.3.5. YouTube

YouTube és un lloc web orientat a compartir vídeos. Els usuaris poden pujar, compartir, veure i comentar vídeos.

Aquesta xarxa dins del projecte té un paper secundari, i dins del mòdul Aplicacions Mòbils només té la funció de mostrar els vídeos de que disposa l'Escola a YouTube.

CAPÍTOL 4. IMPLEMENTACIÓ DEL SERVIDOR

En aquest capítol ens centrarem a descriure la part del servidor "Sóc de l'EETAC" que dóna servei al mòdul d'aplicacions mòbils.

Primer de tot es tractarà quin és el mètode que s'ha utilitzat per l'intercanvi d'informació estructurada entre el servidor i el mòdul d'aplicacions mòbils i com s'ha implementat aquest en el servidor. Seguidament es descriurà cadascun dels objectes i funcions utilitzades pel mòdul d'aplicacions mòbils. Per últim es presentaran les bases de dades temporals utilitzades pel servidor per poder provar el funcionament de l'aplicació mòbil.

4.1. Llenguatge d'intercanvi d'informació estructurada

4.1.1. Escenari

Al mòdul d'aplicacions mòbils conviuen dues plataformes diferents (l'aplicació per a iPhone, plataforma iOS [86] i la plataforma Android). Com ja s'ha comentat, dues plataformes amb dos llenguatges de programació diferents. Per aquest motiu era necessari buscar un sistema d'intercanvi de dades amb el servidor que fos compatible amb les dues plataformes.

El més fàcil era utilitzar un llenguatge senzill de text pla, que fos un estàndard reconegut pels organismes oficials i que fos compatible amb els llenguatges de programació de cada plataforma. Que compleixin amb aquests requisits tenim el llenguatge de marques XML i el llenguatge JSON [92].

4.1.2. XML

XML prové de les sigles en anglès eXtensible Markup Language (llenguatge de marques extensible). És un llenguatge de marques desenvolupat pel World Wide Web Consortium (W3C), el llenguatge permet definir la seva pròpia gramàtica (etiquetes) per estructurar documents.

Una etiqueta consisteix en una marca feta al document XML que senyala una porció del text com un element. Un element és la informació que es vol transmetre. Les etiquetes tenen la forma <etiqueta>, on *etiqueta* és el nom de l'element que s'està senyalant.

```
<?xml version="1.0" encoding="UTF-8" ?>

<Missatge>
        <Remitent>
                <Nom>Nom del remitent</Nom>
                <Correu>Correu del remitent</Correu>
        </Remitent>
```

```
        <Destinatari>
                <Nom>Nom del destinatari</Nom>
                <Correu>Correu del destinatari</Correu>
        </Destinatari>
        <Text>
                <Assumpte>Assumpte del missatgeAssumpte>
                <Cos>Text del missatge<Cos>
        </Text>
</Missatge>
```

Fig. 4.1 Exemple l'estructura d'un document XML

Avantatges:

- Com el seu nom indica, és extensible, un cop dissenyada l'estructura, sempre és possible afegir noves etiquetes.
- Si una tercera part decideix utilitzar un document creat amb XML, és senzill d'entendre la seva estructura i processar-la.
- Es permet verificar que el document estigui ben format i validat

Desavantatges:

- El mapeig de l'estructura d'un document XML cap a altres llenguatges de programació o bases de dades pot ser difícil, sobretot en bases de dades altament estructurades.

4.1.3. JSON

JSON prové de l'acrònim JavaScript Object Notation i és un estàndard obert en text dissenyat per a intercanvi de dades llegible per humans. Deriva del llenguatge JavaScript per representar estructures de dades simples i llistes associatives, anomenades objectes. El llenguatge es descriu a l'RFC 4627.

```
{
"Missatge": {
    "Remitent": {
      "Nom": "Nom del remitent",
      "Correu": "Correu del remitent"
    },
    "Destinatari": {
      "Nom": "Nom del destinatari",
      "Correu": "Correu del destinatari"
    },
    "Text": {
      "Assumpte": "Assumpte del missatge",
      "Cos": "Text del missatge"
    }
  }
}
```

Fig. 4.2 Exemple del format JSON

Avantatges:

- Malgrat la seva relació amb el llenguatge JavaScript, té implementacions per a gran part dels llenguatges de programació.
- JSON pot ser molt compacte i eficient.

Desavantatges:

- La seva breu notació pot ser confusa, especialment quan es tracta de dades fortament jerarquitzades.

4.1.4. Elecció del llenguatge de comunicació estructurada

L'elecció pel llenguatge de comunicació estructurada ha estat JSON degut a tres raons:

- L'aplicació que executa el servidor serà programada amb Java, igual que l'aplicació per a telèfons Android. En aquestes dues aplicacions la conversió d'objectes a JSON des de Java es realitza de forma senzilla i pràcticament nativa.
- Una de les avantatges de JSON és que té implementacions a gairebé tots els llenguatges de programació i l'Objective-C no és una excepció i també el suporta.
- Per últim, es dissenyarà un sistema de comunicació per aplicacions mòbils. Els dispositius actuals disposen de suficient capacitat per rebre o enviar dades gràcies a les xarxes de telefonia mòbil 3G [40] o LTE [55], però seria d'agrair que l'aplicació consumis pocs recursos, i JSON ens permetrà això, ja que la seva notació és més compacte i per tant, haurem d'enviar i rebre menys dades a través de la xarxa que si ho féssim amb XML.

4.1.5. Implementació JSON al Servidor

Un cop escollit el llenguatge de comunicació estructurada, es necessita un altre element per establir aquestes comunicacions, el servei. La millor solució en aquest cas és utilitzar els serveis web basats en REST (RestFul Web Services). Els sistemes operatius per a mòbils poden accedir als serveis web ja que els porten incorporats.

Serà necessari implementar un servei web al servidor i això serà possible a Apache Tomcat [67]. Aquest servei proporciona un entorn on es pot executar codi Java en cooperació amb un servidor web.

Per enviar i rebre dades mitjançant el format JSON necessitarem un mòdul per a l'Apache Tomcat que ens faciliti accedir als serveis web, es tracta del mòdul Jersey.

Jersey té suport per a diferents implementacions de JSON, concretament per a JAXB [90] i Jackson [88]. En el nostre cas s'ha escollit Jackson ja que aquesta implementació ens permetrà enviar llistes d'objectes que després són més fàcils de tractar per les dues aplicacions mòbils.

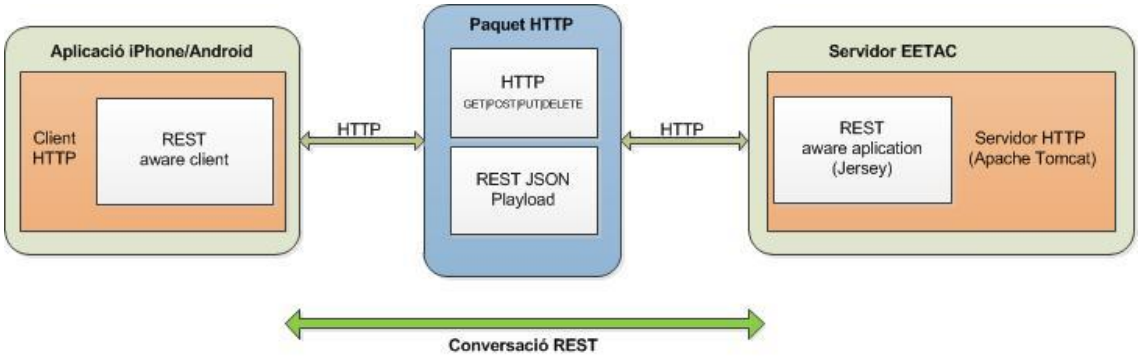

Fig. 4.3 Funcionament complet servei REST

4.1.6. Adreça web d'accés a la informació

L'aplicació mòbil, cada cop que necessiti enviar o rebre informació es connectarà a una URL [102] proporcionada pel servidor Apache Tomcat i el mòdul Jersey, aquesta adreça web està formada per varies parts:

- **IP i port:** la primera part de l'adreça URL conté la IP o el domini i el port del servidor (Apache Tomcat) de l'Escola utilitzat per aquest projecte.
- **Projecte:** aquest part fa referència al nom del projecte: *ImFromEetac*. En el cas que el servidor executi més d'un projecte aquest camp serveix per diferenciar varis projectes.
- **Servei:** aquesta part fa referència al servei que s'utilitza, en el nostre cas: *rest*.
- **Classe:** aquesta part ve determinada per la classe del programa que s'executa al servidor i conté les funcions on s'accediran. En el nostre cas: *ImFromEetac*.
- **Funcions:** aquesta és la última part de la URL, on directament es fa referència a la funció en concreta que volem accedir o executar. En apartats posteriors es definiran les funcions emprades pel projecte.

```
http://147.83.xxx.xxx:8080/ImFromEetac/rest/ImFromEetac/getEetacPerson
```

Fig. 4.4 Exemple d'URL completa

4.2. Objectes

Per a la comunicació entre el servidor i l'aplicació mòbil s'han de definir quins objectes seran necessaris en funció de la informació que es vulgui transmetre i sempre tenint en compte quins camps són necessaris per optimitzar els recursos del telèfon mòbil. Seguidament descriurem els principals objectes que utilitzarà l'aplicació "Sóc de l'EETAC" juntament amb el servidor, i quin tipus de dades conten cadascun d'aquests objectes.

4.2.1. EetacPerson

L'objecte EetacPerson és el principal element d'aquestes comunicacions. S'utilitzarà en varis apartats, ja sigui enviant la informació completa dels seus camps o només de forma parcial. Disposa de quatre tipus de camps ben diferenciats.

4.3.1.1. Camps referents a la informació proporcionada per l'EETAC

Aquests camps bàsicament són els camps que ens transmet el mòdul Bus SOA i que contenen la informació de cadascun dels estudiant de l'EETAC:

- ***estudiantEetac (String [100])*:** conté el nom i cognoms de l'estudiant, amb el següent format: *COGNOMS, NOM*.
- ***emailEstudiantEetac (String)*:** conté l'adreça de correu electrònic de l'Escola, amb el següent format: *nom.cognom@estudiant.upc.edu*.
- ***quadrimestreIngresEetac (String)*:** conté el quadrimestre que l'estudiant va ingressar a l'Escola, amb el següent format: *any-quadrimestre* amb números.
- ***quadrimestreObtencioTitulacio (String)*:** conté el quadrimestre on l'estudiant va finalitzar la seva titulació, amb el següent format: *any-trimestre* amb números.
- ***titulacio (String)*:** nom de la titulació que cursa o ha cursat l'estudiant, amb el següent format: *ENG TELEC 2NCICLE 01, AERO/ETT SIST 05, ET AERONÀUTICA 03, ETT SIST TELEC 00, ETT SIST TELEC 91, ETT SIST TELEC 91, ETT SIST/ ET AERO 05, ETT TELEMÀTICA 00, MO AEROSPACE S&T, MU MASTEAM*.
- ***dataTitulacio (String)*:** conté la data exacta de quan l'estudiant va obtenir la titulació, amb el següent format: *dd/mm/aaaa*.

4.3.1.2. Camps referents a la informació proporcionada per l'Aplicació

Aquest camps són els que proporciona la pròpia aplicació "Sóc de l'EETAC al servidor i són els següents:

- ***eetacID (String)*:** identificador únic per a cada estudiant que utilitzi l'aplicació. Simplifica la identificació de les seves dades en el servidor.
- ***lat* i *lon (double [76])*:** proporcionen la latitud i longitud de l'estudiant que utilitza l'aplicació.
- ***twitterName (String)*:** proporciona al servidor l'identificador de la compte de Twitter de l'usuari de l'aplicació en el cas que aquest l'activi.

4.3.1.3. Camps referents a la informació proporcionada per LinkedIn

Aquests camps els proporciona la xarxa professional de LinkedIn, principalment proporciona el perfil i les dades laborals de cada estudiant:

- ***linkedInID (String)*:** conté l'identificador únic de cada usuari de LinkedIn. Permetrà al servidor relacionar i actualitzar les dades laborals de cada estudiant.
- ***name (String)*:** conté el nom de l'usuari de LinkedIn.
- ***surname (String)*:** conté els cognoms de l'usuari de LinkedIn.
- ***urlImageLinkedIn*:** conté l'enllaç de la imatge utilitzada per l'usuari a LinkedIn, si en disposa.
- ***currentJobs (List<Job> [94])*:** conté la llista de les feines actuals de l'usuari de LinkedIn.
- ***pastJobs (List<Job>)*:** conté la llista de les feines anteriors de l'usuari de LinkedIn.

Els camps *correntJobs* i *pastJobs* contenen una llista d'objectes del tipus *Job*. *Job* és un altre objecte que conté els principals camps d'una feina:

- ***startDateJob (String)*:** data d'inici d'una feina.
- ***endJobDate (String)*:** data de finalització d'una feina.
- ***companyPosition (String)*:** càrrec ocupat en aquesta feina.
- ***company (Company)*:** conté informació sobre l'empresa on s'ha realitzat la feina.

Igual que abans el camp *company* és un objecte que conté camps relacionats amb l'empresa i són del tipus:

- ***companyName (String)*:** conté el nom d'una empresa.
- ***companyDescription (String)*:** conté una descripció o categoria d'una empresa.
- ***companyAddress (String)*:** conté l'adreça d'una empresa.
- ***companyLat i companyLon (double)*:** conté la latitud i la longitud d'una empresa, servirà per posicionar l'empresa en un mapa.

4.3.1.4. Camps referents als permisos de l'usuari

Aquests camps contenen el permisos necessaris per compartir la informació amb altres usuaris de l'aplicació. Aquest camps restringeixen o donen accés a cada tipus d'informació descrita anteriorment i són de tipus booleà:

- ***allowEmailEstudiantEetac (Boolean [72])*:** dóna o restringeix el permís per compartir el correu electrònic de l'Escola.
- ***allowTitulacio (Boolean)*:** dóna o restringeix el permís per compartir la titulació de l'estudiant.
- ***allowDataTitulació (Boolean)*:** dóna o restringeix el permís per compartir la data exacta de l'obtenció de la titulació.
- ***allowAcces (Boolean)*:** dóna o restringeix l'accés a l'aplicació i a les dades de cada estudiant.
- ***allowLocation (Boolean)*:** dóna o restringeix el permís per compartir la localització d'un estudiant.
- ***allowMessages (Boolean)*:** dóna o restringeix el permís per compartir missatges a través de l'aplicació.
- ***allowName (Boolean)*:** dóna o restringeix el permís per mostrar el nom del perfil de LinkedIn.
- ***allowSurname (Boolean)*:** dóna o restringeix el permís per mostrar els cognoms del perfil de LinkedIn.
- ***allowImageLinkedIn (Boolean)*:** dóna o restringeix el permís per mostrar la imatge del perfil de LinkedIn
- ***allowCurrentJobs (Boolean)*:** dóna o restringeix el permís per compartir el llistat de feines actuals de l'usuari de LinkedIn.
- ***allowPastJobs (Boolean)*:** dóna o restringeix el permís per compartir el llistat de feines anteriors de l'usuari de LinkedIn.

4.2.2. ImHere

Aquest objecte serà necessari quan s'hagi d'enviar la posició de l'usuari de l'aplicació o rebre informació sobre la posició d'algun altre estudiant. L'objecte disposa dels següents camps:

- ***eetacID (String)*:** aquest camp servirà per identificar les dades d'un usuari en concret.
- ***myLon* i *myLat (double [76])*:** conté la longitud i latitud de la posició de l'usuari de l'aplicació o el centre de l'àrea d'on es vol localitzar altres estudiants.
- ***mapDistance (double)*:** conté el radi per determinar l'àrea de cerca d'altres estudiants.
- ***allowPosition (Boolean)*:** indica si compartim la posició de l'usuari o el centre de l'àrea.

4.2.3. Message

Aquest objecte serà necessari quan s'intercanviïn missatges amb altres usuaris de l'aplicació i contindrà tota la informació necessària perquè els missatges es distribueixin correctament. Els seus camps són:

- ***messageID (int)*:** identificador numèric d'un missatge.
- ***eetacID (String)*:** identificador únic de l'estudiant que envia el missatge.
- ***twitterName (String)*:** identificador de Twitter de l'estudiant que envia el missatge.
- ***student (String)*:** nom de l'estudiant que envia el missatge.
- ***image (String)*:** adreça web de la imatge de LinkedIn de l'estudiant que envia el missatge.
- ***destinationEetacID (String)*:** identificador únic de l'estudiant a qui va dirigit el missatge.
- ***destinationTwitterName (String)*:** identificador de Twitter de l'estudiant a qui va dirigit el missatge.
- ***destinationStudent (String)*:** nom de l'estudiant a qui va dirigit el missatge.
- ***destinationImage (String)*:** imatge de LinkedIn de l'estudiant a qui va dirigit el missatge.
- ***subject (String)*:** assumpte del missatge.
- ***date (String)*:** data de la creació del missatge.
- ***message (String)*:** text i cos del missatge.

L'objecte Message també va acompanyat d'uns altres objectes que fan possible diferents operacions que es poden realitzar amb els missatges, per exemple llistar els missatges enviats o rebuts, o eliminar un missatge. Es tracta dels objectes *RequestMessages*, *EetacIDMessages* i *Messages*.

4.2.4. WebsInput

Aquest objecte serà l'encarregat de contenir la informació necessària per llistar les diferents webs de serveis i informació de l'EETAC. Els seus camps són:

- ***urlImage (String)*:** adreça web que conté la imatge que de la pàgina web.
- ***description (String)*:** text descriptiu de la pàgina web.
- ***webURL (String)*:** adreça de la pàgina web que volem accedir.

L'objecte *WebsInput* també va acompanyat d'un altre objecte que agrupa diferents entrades de webs en un tema i les classifica per títols, es tracta de l'objecte *WebsGroups*.

4.2.5. StatisticsInput

Aquest objecte serà l'encarregat de llistar les diferents entrades de dades i estadístiques referents als estudiants i estudis de l'EETAC. Els seus camps són els següents:

- ***title (String)*:** títol de les dades a mostrar.
- ***shortDescription (String)*:** descripció curta del tema de les dades.
- ***description (String)*:** text complet de les dades a mostrar.
- ***statisticsURL (String)*:** adreça de la imatge que conté la gràfica de les dades a mostrar.

L'objecte *StatisticsInput* també va acompanyat d'un altre objecte que agrupa diferents entrades de dades en un tema i les classifica per títols, es tracta de l'objecte *StatisticsGroups*.

4.3. Funcions

En aquest apartat es veurà quines funcions s'han implementat al servidor per a la comunicació de dades amb l'aplicació mòbil i quins objectes han intervingut en elles.

S'han utilitzat dos tipus de funcions per a dues necessitats diferents referents a la comunicació de dades entre l'aplicació i el servidor. A la primera, l'aplicació només necessita rebre dades sense haver de preguntar quines són aquestes dades, ja que aquestes són de caràcter fixe, aquestes funcions utilitzaran el mètode GET [79] del servei web. A la segona, l'aplicació necessita enviar certs paràmetres per obtenir les dades desitjades i aquestes funcions utilitzaran el mètode POST [97]. L'aplicació també necessita enviar dades al servidor i també es farà a través del mètode POST del servei web, ja que el servidor després ens retornarà una resposta amb el resultat de l'operació realitzada.

També hi ha un seguit de funcions que s'han utilitzat per provar i corregir resultats d'operacions de l'aplicació mòbil contra la base de dades del servidor. Aquestes funcions han sigut importants per comprovar el funcionament i la lògica de l'aplicació mòbil.

4.3.1. Funcions amb el mètode GET

4.3.1.1. getStatisticsList

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil un llistat de dades i estadístiques amb informació referent als estudiant i estudis de l'Escola.

En aquesta funció hi intervenen els objectes *StatisticsGroups* i *StatisticsInput*. El primer agrupa diferents *StatisticsInput* en un tema i el segon conté la informació concreta de les estadístiques a mostrar a l'aplicació.

4.3.1.2. getWebList

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil un llistat de pàgines web de l'EETAC que contenen els serveis i informació de l'Escola.

En aquesta funció hi intervenen els objectes *WebsGroups* i *WebsInput*. El primer agrupa diferents objectes *WebsInput* en un tema i el segon conté la informació concreta a mostrar a l'aplicació, en aquest cas l'adreça de la pàgina web.

4.3.1.3. getMenuBackgrounds

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil els fons de pantalla que apareixen al menú amb els seus enllaços corresponents.

En aquesta funció hi intervé l'objecte *BackgroundPicture* i s'envia a l'aplicació un llistat amb aquest tipus d'objecte.

4.3.2. Funcions amb el mètode POST

4.3.2.1. getEetacPerson

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil la informació d'un estudiant de l'EETAC, ja sigui de forma completa o parcial.

En aquesta funció hi intervé l'objecte *EetacPerson* que és l'encarregat de contenir la informació concreta d'un estudiant. El servidor envia directament aquest objecte o una llista d'*EetacPerson*. També hi intervé l'objecte *NewID*, aquest objecte l'envia l'aplicació mòbil i és l'encarregat de determinar quin objecte *EetacPerson* volem rebre.

4.3.2.2. updateEetacPerson

Aquesta funció és l'encarregada de rebre de l'aplicació mòbil la informació d'un estudiant de l'EETAC per actualitzar-la a la base de dades.

En aquesta funció hi intervé l'objecte *EetacPerson,* l'envia l'aplicació mòbil i conté tota la informació recollida per l'aplicació de l'estudiant. També hi intervé

l'objecte *Resposta*, tal com indica el seu nom, el servidor envia aquest objecte amb el resultat l'operació, en aquest cas, si la informació de l'estudiant s'ha actualitzat.

4.3.2.3. getLocalitzationEetacPeople

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil la informació d'un o varis estudiant de l'EETAC que apareguin a l'àrea del mapa de l'apartat Companys.

En aquesta funció hi intervé l'objecte *EetacPerson* que és l'encarregat de contenir la informació concreta d'un estudiant i el servidor és l'encarregat de buscar i enviar una llista d'aquest objectes si es tenen companys a prop. També hi intervé l'objecte *ImHere*, aquest objecte l'envia l'aplicació mòbil i és l'encarregat de definir qui és l'usuari de l'aplicació i quina és l'àrea del mapa on s'estan buscant altres companys de l'EETAC.

4.3.2.4. updateLocationEetacPerson

Aquesta funció és l'encarregada de rebre de l'aplicació mòbil la informació de la localització d'un estudiant de l'EETAC.

En aquesta funció hi intervé l'objecte *ImHere,* aquest objecte l'envia l'aplicació mòbil i és l'encarregat de definir la posició de l'usuari de l'aplicació. També hi intervé l'objecte *Resposta*, el servidor envia aquest objecte amb el resultat l'operació, en aquest cas, si la posició de l'usuari de l'aplicació s'ha actualitzat.

4.3.2.5. getListEetacPeopleMessages

Aquesta funció és l'encarregada d'enviar a l'aplicació mòbil la llista de companys que utilitzen el servei de missatgeria.

En aquesta funció hi intervé l'objecte *EetacPerson* que és l'encarregat de contenir la informació concreta d'un estudiant. El servidor és l'encarregat de buscar quins uns usuaris utilitzen el servei de missatgeria i enviar una llistat d'aquests objectes. L'aplicació també envia un objecte *EetacPerson* per determinar qui sol·licita aquesta llista d'usuaris.

4.3.2.6. getSendMessages i getReceivedMessages

Aquestes funcions són les encarregades d'enviar a l'aplicació mòbil la llista de missatges enviats i rebuts respectivament per l'usuari de l'aplicació al servei de missatgeria.
En aquestes funcions hi intervé l'objecte *Messages* que conté la llista de missatges enviats i rebuts respectivament per l'usuari de l'aplicació. L'objecte *Messages* està compost per una llista d'objectes *Message*, aquest conté la informació concreta d'un missatge. L'aplicació també envia un objecte *RequestMessages* per determinar qui sol·licita aquestes llistes de missatges i quins missatges s'enviaran dins de la llista.

4.3.2.7. setMessage

Aquesta funció és l'encarregada de rebre de l'aplicació mòbil un missatge enviat des del servei de missatgeria.

En aquesta funció hi intervé l'objecte *Message* que conté la informació d'un missatge. També hi intervé l'objecte *Resposta*, el servidor envia aquest objecte amb el resultat l'operació, en aquest cas, si el missatge s'ha guardat correctament.

4.3.2.8. deleteFromSendMessages i deleteFromReceivedMessages

Aquestes funcions són les encarregades d'eliminar un missatge de la llista de missatges enviats o rebuts respectivament.

En aquesta funció hi intervé l'objecte *RequestMessages* que conté l'identificador del missatge que es vol eliminar. També hi intervé l'objecte *Resposta*, el servidor envia aquest objecte amb el resultat l'operació, en aquest cas, si el missatge s'ha eliminat correctament de la llista de missatges enviats o rebuts respectivament.

4.3.3. Funcions de testeig i control

4.3.3.1. serverIsRunning

Aquesta funció determina l'estat del servidor, utilitza el mètode GET [79] del servei web i és l'encarregada d'enviar un objecte *Resposta* amb el valor "Sí". Serveix per comprovar ràpidament si el servidor està funcionant i envia informació correctament.

4.3.3.2. existsEetacPerson

Aquesta funció retorna un objecte *Resposta* amb la resposta de l'existència d'un usuari i utilitza el mètode POST. L'aplicació envia un objecte *EetacPerson* per determinar si l'estudiant que es vol comprovar existeix a la base de dades.

4.3.3.3. getDataBase

Aquesta funció retorna una llista amb les dades de tots els estudiants, utilitza el mètode GET i és l'encarregada d'enviar un llistat d'objectes *EetacPerson*.

4.3.3.4. getDataBaseMessages

Aquesta funció retorna una llista amb tots els missatges enviats i rebuts de cada estudiant, utilitza el mètode GET i és l'encarregada d'enviar un llistat d'objectes del tipus *EetacIDMessages*, aquest objecte conté les llistes de missatge enviats i rebuts d'un sol estudiant.

4.4. Bases de dades temporals

En aquest apartat veurem quines bases de dades s'han creat per poder comprovar el funcionament de l'aplicació mòbil. Aquestes bases de dades s'han implementat al servidor de forma no persistent, ja que aquesta no és la finalitat, ni l'objectiu d'aquest projecte. L'encarregat del funcionament complet del servidor el tractarà el projecte Sóc de l'EETAC: Desenvolupament d'una eina de recollida d'informació a les xarxes professionals de titulats/des.

4.4.1. Implementació

Aquestes bases de dades s'han implementat utilitzant un *Singleton* diferent per a cadascuna d'elles.

Un *Singleton [29]* és una instància única per a una classe i està dissenyat per restringir la creació d'objectes pertanyents a aquesta classe o restringir el valor d'un tipus d'objecte. La seva intenció és garantir que una classe només tingui una instància i proporcioni un punt d'accés global a ella. Aquesta última característica ens permetrà utilitzar el Singletó com a base de dades, ja que dins de cada classe s'introduiran els objectes i les dades necessàries per al funcionament de l'aplicació mòbil, i des de qualsevol punt del programa es podrà accedir a aquestes dades, crear-ne de noves, modificar-les o eliminar-les.

Aquest mètode té una desavantatge respecte altres bases de dades, té un comportament no persistent, això vol dir, que cada cop que es s'executi de nou l'aplicació del servidor, els canvis introduïts a les bases de dades s'eliminaran i tornaran a tenir els valors per defecte definits en un inici.

4.4.2. Bases de Dades

4.4.2.1. DBManger

Aquesta base de dades és l'encarregada d'administrar tota la informació de cada estudiant de l'EETAC, està formada per una multitud d'objectes *EetacPerson*.

4.1.1.1. DBMangerMessages

Aquesta base de dades és l'encarregada d'administrar tots els missatges enviats i rebuts per cada estudiant en el servei de missatgeria de l'aplicació mòbil, està formada per una multitud d'objectes *Message*.

4.1.1.2. WebsDBManger

Aquesta base de dades és l'encarregada d'administrar totes les pàgines web que contenen informació i serveis de l'EETAC, està formada per un conjunt d'objectes *WebsInput* i *WebsGroups*.

4.1.1.3. StatisticsDBManger

Aquesta base de dades és l'encarregada d'administrar totes les dades i estadístiques que l'EETAC vol mostrar a l'aplicació mòbil, està formada per un conjunt d'objectes *StatisticsInput* i *StatisticsGroups*.

4.1.1.4. BackgroundsDBManger

Aquesta base de dades és l'encarregada d'administrar els diferents fons de pantalla que es mostren a l'aplicació mòbil, està formada per un conjunt d'objectes *BackgroundPicture*.

CAPÍTOL 5. IMPLEMENTACIÓ DE L'APLICACIÓ

En aquest apartat presentarem el funcionament i la implementació de cadascun dels mòduls de l'aplicació "Sóc de l'EETAC". Primer de tot es veuran quins apartats s'han dissenyat. A continuació es descriuran les llibreries utilitzades a l'aplicació, tant les llibreries externes desenvolupades per terceres parts com les pròpies llibreries de cada plataforma. Seguidament es descriurà com s'implementen i es creen les vistes, tant per Android [66] com per iOS [86]. I per últim presentarem i veurem el funcionament i implementació de les principals vistes de l'aplicació.

5.1. Disseny i elecció dels mòduls de l'aplicació

Seguint els objectius inicials, primer de tot es va decidir que faria falta un sistema d'autentificació per comprovar si l'usuari de l'aplicació realment havia estat estudiant de l'EETAC. Aprofitant que un dels objectius era facilitar la recollida d'informació laboral d'antics alumnes de l'EETAC a través de la xarxa professional de LinkedIn, i que seria necessari que l'usuari de l'aplicació s'identifiqués per obtenir les seves dades laborals públiques, es va decidir aprofitar aquest sistema d'identificació de LinkedIn per comprovar i validar l'usuari de l'aplicació a la base de dades de l'Escola. El resultat és la vista d'Accés a l'aplicació. A l'aplicació també s'hi podrà accedir com a convidat amb certes restriccions d'ús.

Per facilitar la comunicació dels antics alumnes a través d'un servei de missatgeria. Seguint aquest objectiu es va dissenyar l'apartat Missatges en el qual es podrà enviar i rebre missatges a través d'un servei propi o utilitzant la xarxa social de Twitter.

Per geoposicionar els usuaris de l'aplicació i els seus llocs de treball. Seguint aquest objectiu és va dissenyar l'apartat Companys amb un mapa, amb la localització dels usuaris i mostrant les seves dades i amb la possibilitat d'enviar missatges, tot això utilitzant un sistema de permisos per compartir les dades.

Per visualitzar els continguts de les xarxes socials que utilitza l'Escola: LinkedIn, Twitter i interactuar amb elles. D'aquest objectiu neix l'apartat Social mostrant d'una forma ràpida aquestes xarxes socials a l'usuari de l'aplicació i amb l'opció d'interaccionar amb elles (recomanar, comentar o repiular [99]).

Per visualitzar tot un seguit de dades i estadístiques referents a l'Escola per promocionar-la, neix l'apartat de Dades, i també l'apartat L'Escola amb informació sobre estudiants, serveis i estudis de l'Escola. De manera que l'aplicació canvia l'ús original de ser una aplicació per a antics alumnes a ser una aplicació per a futurs, actuals i antics alumnes de l'EETAC, on cada usuari podrà treure un profit diferent d'aquesta aplicació.

Finalment, tota aplicació té un apartat on es poden gestionar diferents opcions. En aquest apartat es podrà gestionar quines dades es comparteixen amb altres usuaris i es podrà desvincular o tornar a vincular l'aplicació a les xarxes socials de LinkedIn i Twitter. També mostrarà un resum de les dades que l'usuari té tant de l'Escola, com de LinkedIn, com de Twitter.

Ja tenim l'aplicació "Sóc de l'EETAC" dissenyada, amb un sistema d'Accés i un Menú amb sis apartats ordenats de la següent manera: L'Escola, Missatges, Companys, Social, Dades i Opcions.

5.2. Llibreries Externes

A continuació es presentaran les llibreries externes que es faran servir sobre els diferents sistemes operatius (Android, iOS). Per al sistema operatiu Android s'explicarà l'ús de llibreries per serialitzar dades (com Gson), bases de dades embebides (db40), llibreria d'accés a les APIs de Twitter (twitter4j), LinkedIn (linkedin-j). Per iOS s'explicarà l´ús de la llibreria d'accés a la API de LinkedIn (Results Direct LinkedIn iPhone).

5.2.1. Android

5.2.1.1. Gson

Gson és una llibreria Java [89] que es pot utilitzar per convertir objectes Java a la seva representació JSON [92] i a l'inrevés. Els objectius de Gson són:

- Permetre la conversió entre objectes Java i JSON d'una manera senzilla, simplement invocant els mètodes *toJson()* o *fromJson()*.
- Permetre la conversió d'objectes immutables ja existents.
- Suportar tipus genèrics de Java.
- Permetre la representació personalitzada d'objectes.
- Suportar "Objectes arbitràriament complexos".

Per a la realització del projecte, ha estat de gran ajuda, ja que totes les comunicacions realitzades amb el servidor són serialitzades en format JSON. La informació intercanviada amb el servidor ha estat en continua evolució, afegint i eliminant paràmetres, però gràcies a aquesta llibreria no ha estat necessari modificar les diferents parts de codi on es serialitzaven i es deserialitzaven els objectes.

Per serialitzar un objecte es crea una variable del tipus *Gson*, i tot seguit, utilitzant aquesta variable es serialitza per exemple un objecte del tipus *EetacPerson* i es guarda a la variable *serialized*, que és de tipus *String*.

```
Gson Gson = new Gson();
serialized = Gson.toJson(eetacPerson);
```

Fig. 5.1 Exemple de serialització utilitzant Gson

Per deserialitzar un objecte, el procés és molt semblant i igual de senzill. Ara l'objecte *EetacPerson* està buit, i per omplir-lo, el que fem, és passar-li dos paràmetres a Gson. El primer és la informació serialitzada en format JSON, *serialized*; i el segon, indica en quin format han d'anar les dades, en aquest cas han de ser deserialitzades com un objecte *EetacPerson*.

```
EetacPerson eetacPerson = Gson.fromJson(serialized, EetacPerson.class);
```

Fig. 5.2 Exemple de deserialització utilitzant Gson

Tot i la facilitat d'ús de la llibreria, durant el procés de desenvolupament de l'aplicació hi hagut certes dificultats causades per la mateixa, ja que a l'hora de deserialitzar la informació procedent del servidor, en alguns casos donava errors a les capçaleres.

5.2.1.2. GreenDroid

GreenDroid és una llibreria de desenvolupament per a la plataforma Android, la seva finalitat és poder realitzar desenvolupaments d'interfície d'usuari més fàcils i consistents. Aquesta llibreria, entre altres coses, ens permet afegir un *ActionBar* a les nostres aplicacions, per donar-los un aspecte més formal i fer que siguin una mica més agradables a la vista, a part de funcionals.

A l'aplicació, l'ús d'aquesta llibreria és molt visible, ja que, pràcticament en tots els apartats es pot observar una barra de color blau a la part superior.

Fig. 5.3 *ActionBar* utilitzant GreenDroid

La seva utilitat no radica en col·locar la barra a la part superior, operació relativament senzilla d'implementar mitjançant els mètodes tradicionals, sinó en el fet d'afegir i treure botons, i col·locar submenús a cada boto. L'opció "torna a

casa" també es realitza de forma senzilla, representada amb la icona d'una casa.

Una altre de les funcionalitats de la llibreria que ha estat inclosa en el projecte, és la capacitat de col·locar pestanyes.

Fig. 5.4 Ús de pestanyes utilitzant GreenDroid

El primer que cal fer, per començar a utilitzar la barra és substituir a totes les *Activitys [65]*, la herència de les mateixes, així en comptes de heretar de *Activity* com seria el més habitual, ara heretarà de *GDActivity*. A la funció *onCreate*() a l'hora de relacionar l'Activity amb un *Layout [93]*, també pateix un petit canvi, modificant la crida.

```
public class Webs extends Activity {
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.list_adapter);
```

Fig. 5.5 Exemple sense GreenDroid

```
public class Webs extends GDActivity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setActionBarContentView(R.layout.list_adapter);
```

Fig. 5.6 Exemple amb GreenDroid

Per poder afegir botons a la barra, GreenDroid [80] té una classe de tipus de botons, amb la qual cosa no ens haurem de preocupar per dissenyar les icones, només s'haurà de seleccionar-ne una de la classe *Type*. Cal tenir en compte que la barra no admet més de tres icones.

Ara ens falta crear un objecte *application*, per ser més precisos, un objecte que hereti de *GDApplication* i que implementi el mètode *getHomeActivityClass*() que servirà per saber quan l'aplicació ha de mostrar el botó de "torna a casa", per retorna al menú principal.

Ara queda definir l'aspecte de la barra, serà de color blau, respectant la tonalitat del logotip de l'Escola, amb les lletres blanques. S'ha de crear un nou arxiu *XML* [103] anomenat *theme.* Aquest arxiu es col·locarà dins de la carpeta *res\values*. A l'interior d'aquest arxiu és on es defineix el disseny de la barra.

```
<?xml version="1.0" encoding="utf-8"?>
 <resources>
 <style name="Theme.Mytheme" parent="@style/Theme.GreenDroid.NoTitleBar">
   <item name="gdActionBarTitleColor">@android:color/white</item>
   <item name="gdActionBarDividerWidth">2px</item>
   <item name="gdActionBarBackground">#00a4f9 </item>
   <item name="gdActionBarApplicationDrawable">@drawable/logotip</item>
 </style>
```

Fig. 5.7 Contingut de l'arxiu XML *theme*

5.2.1.3. DB4O

DB4O és una llibreria de dades de codi obert que permet als desenvolupadors de Java i .NET guardar i recuperar qualsevol objecte amb una sola línia de codi, eliminant la necessitat de predefinir o mantenir un model de dades rígid. Les seves sigles es corresponen amb l'expressió "DataBase 4 (for) Objects".

Les claus d'aquesta llibreria són el seu alt rendiment i el model de desenvolupament que proporciona a les aplicacions per la seva capa d'accés a dades.

En aquest cas, l'aplicació ha de permetre emmagatzemar objectes, i objectes compostos de diversos objectes més. Això en DB4O, és tant transparent com persistir l'objecte principal. Per guardar objectes a la base de dades s'escriu la següent línia de codi:

```
ObjectData.getInstance(this.getApplicationContext()).saveObject(person);
```

Fig. 5.8 Guarda l'objecte *person* a la base de dades

```
EetacPerson person = (EetacPerson) ObjectData.getInstance
(this.getApplicationContext()).loadObject(EetacPerson.class);
```

Fig. 5.9 Extreu informació de la base de dades

Cal afegir, que s'utilitza la llibreria DB4O només per guardar informació de l'usuari de l'aplicació. Aquest és el motiu per el qual es necessita buscar informació dins de la base de dades. Cada vegada que es requereix informació s'extreu tot el contingut. Per actualitzar-la es substitueix tota la informació.

5.2.1.4. Twitter4j

Twitter4j és una llibreria que facilita la connexió amb la *API [68]* de Twitter, ajudant d'aquesta manera a integrar l'aplicació amb els serveis que ofereix l'esmentada xarxa social.

Tant a Twitter com a LinkedIn la llibreria necessita uns *Access Tokens,* que s'obtenen amb l'ajuda de la llibreria de *SingPost.*

Les funcions principals que ajuda a resoldre la llibreria són:

- Identificar un usuari.
- Descarregar el llistat de piulades (*Timeline*) que ofereix l'Escola així com l'opció de repiular-los [99].
- Enviar i rebre piulades amb el *hashtag [82] #socdelEETAC*.

Per aconseguir el *timeline* de l'Escola ho fem utilitzant la següent línia de codi:

```
List<Status> homeTimeline = twitter.getUserTimeline(UserTimeline);
```

Fig. 5.10 Descarrega de les piulades realitzades per l'Escola

On *UserTimeline* és una variable de tipus *String* que conté el nom d'usuari de Twitter que utilitza l'Escola (@EETAC_UPC).

L'opció de realitzar una repiulada, és més complexa que aconseguir el llistat de tuits, ja que requereix que l'usuari estigui autentificat. La autentificació requereix, per una banda, de les claus *consumer key* i *consumer secret* proporcionades a la web de Twitter developers; i per l'altre banda, de les claus *Access token* i *Acess token secret* obtingudes al vincular l'aplicació amb el Twitter. Un cop identificats ja podem realitzar la repiulada.

```
twitter.setOAuthConsumer(consumer[0], consumer[1]);
accessToken = new AccessToken(tokens[0], tokens[1]);
twitter.setOAuthAccessToken(accessToken);
```

Fig. 5.11 Autentificació amb el Twitter

```
twitter.retweetStatus(homeTimeline.get(position).getId());
```

Fig. 5.12 Realització d'una repiulada

La última funció que realitza la llibreria, és la de realitza tuits mencionant el *hashtag #socdelEETAC* juntament amb el nom de l'usuari a qui es vulgui mencionar. Com al cas de repiular, per a la realització d'un tuit [101] cal realitzar una identificació de l'usuari prèvia al missatge, i tot seguit ja es pot enviar.

```
new TwitterClass().sendMessage(getApplicationContext(), message);
```

Fig. 5.13 Realització d'un tuit

5.2.1.5. LinkedIn-j

La llibreria de LinkedIn-j té una utilitat i funcionament similar a la de Twitter. Les funcions que ens ajuda a resoldre són:

- Identificar un usuari.
- Recollir informació de l'usuari identificat.
- Descarregar el llistat de discussions (*Timeline*) que ofereix l'Escola així com l'opció de recomanar-les.

La identificació requereix, a l'igual que a Twitter, d'un seguit de claus tant per identificar l'aplicació com per identificar al usuari. Les claus per identificar l'aplicació s'obtenen des de la web de desenvolupadors de LinkedIn i les d'usuari s'obtenen al vincular l'aplicació amb la seva compte.

Per recollir informació sobre l'usuari, a la llibreria se li hauran de passar els paràmetres que es necessiten per obtenir el perfil, en altre paraules, només es descarrega la informació que es necessita per l'aplicació, no el perfil complert.

```
LinkedInAccessToken accessToken = new LinkedInAccessToken(tokens[0], tokens[1]);
client = factory.createLinkedInApiClient(accessToken);

personLi = client.getProfileForCurrentUser(EnumSet.of(ProfileField.FIRST_NAME,
ProfileField.LAST_NAME, ProfileField.PICTURE_URL, ProfileField.ID,ProfileField.POSITIONS));
```

Fig. 5.14 Obtenció del perfil de LinkedIn

La informació que es descarrega de LinkedIn és, el nom i cognom, la URL de la imatge d'usuari, el ID (identificador únic per cada usuari) i els llocs de treball, tant l'actual com els anteriors, la llibreria els anomena *Positions.*

Tota aquesta informació es guarda a l'objecte creat per la llibreria *personLi,* que és del tipus *Person*. Un cop aconseguida aquesta informació cal tractar-la per guarda-la a la base de dades.

Per aconseguir el llistat de discussions, a diferència de Twitter, cal que l'usuari estigui identificat prèviament, un cop identificat ja es pot procedir a la descàrrega del llistat.

```
final LinkedInApiClientFactory factory = LinkedInApiClientFactory.newInstance(li[0], li[1]);
LinkedInAccessToken accessToken = new LinkedInAccessToken(tokens[0], tokens[1]);
client = factory.createLinkedInApiClient(accessToken);
group = client.getGroupById("86759", EnumSet.of(GroupField.SMALL_LOGO_URL,GroupField.NAME));
Posts posts = client.getPostsByGroup("86759", EnumSet.of(PostField.ID,
PostField.CREATION_TIMESTAMP,PostField.CREATOR_LAST_NAME, PostField.CREATOR_FIRST_NAME,
PostField.SUMMARY, PostField.CREATOR_PICTURE_URL,PostField.TITLE));
```

Fig. 5.15 Obtenció de les discussions de LinkedIn

Mitjançant l'objecte *LinkedInAccessToken,* s'identifica l'usuari amb els *tokens* aconseguits prèviament. El mecanisme per rebre la informació del grup és similar al de rebre la informació d'usuari. El primer paràmetre que es necessita, és l'identificador de grup. A l'igual que l'identificador de l'usuari, aquest també és únic. Després s'indicarà quina és la informació que es vol extreure. Per al cas del grup, es necessita la URL de la imatge i el nom del grup.

Per obtenir el llistat de discussions, es fa ús de la funció *getPostsByGroup*, i la dinàmica torna a ser la mateixa, se li ha de passar l'identificador de grup i la informació que es desitja rebre. Per al cas de les discussions es necessita l'identificador del post, per poder recomanar en cas necessari, l'hora de la creació, el nom i cognom del creador, el cos de la discussió, la adreça web de la imatge del creador i per acabar, el títol.

La opció de recomanar requereix l'identificador de cada discussió, i que l'usuari estigui identificat. Tant la identificació, com el fet d'aconseguir el identificador de cada discussió ja es realitza prèviament, al rebre el llistat de discussions. D'aquesta manera només cal fer la recomanació en funció de la posició que ocupa la discussió al llistat que es mostra per pantalla.

```
client.likeGroupPost(post.get(position).getId());
```

Fig. 5.16 Recomanar una discussió

5.2.1.6. Signpost

Signpost és una solució fàcil i intuïtiva per registrar-se utilitzant missatges HTTP [84] mitjançant el protocol obert OAuth, que permet una autorització segura. La llibreria ha estat utilitzada conjuntament amb les de Twitter i LinkedIn. Ha sigut necessària en el moment en que l'aplicació demana a l'usuari accés a les respectives comptes de xarxes social.

5.2.1.7. Android MapView Balloons

Aquesta llibreria fa possible, de forma senzilla, la col·locació de globus amb informació extra sobre els mapes de Google, és molt útil ja que aquesta funció no està contemplada dins de l'API [68] de Google Maps. A part de mostrar un globus sobre una posició del mapa, també fa que el mateix globus sigui clicable, permeten saltar a una altra vista per ampliar la informació mostrada.

Tot i la seva utilitat, la llibreria ha estat modificada lleugerament per adaptar-la a les necessitats de l'aplicació. El problema que presentava, era que per defecte la llibreria posava el globus al punt central de les coordenades rebudes, i coincidia amb la part baixa de la figura on es marca la posició d'un titulat, i el prémer sobre el titulat el globus el tapava.

Un cop modificada la llibreria, el resultat de la posició del globus s'ha vist desplaçada cap a la part superior de la figura en comptes dels peus.

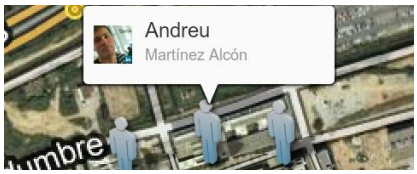

Fig. 5.17 Detall de la llibreria MapView Balloon

Cal crear una iteració que per cada persona que es rebi, al prémer a la seva posició mostra un globus, al globus se li passen diferents paràmetres com ara la posició (*geoPoint*), per saber a on s'ha de mostrar, el nom i el cognoms (*temp.getName* i *temp.getSurname),* i la adreça web de la imatge (*temp.getUrlImageLinkedIn)*.

```
CustomOverlayItem overlayItem = new
CustomOverlayItem(geoPoint,temp.getName(),temp.getSurname(), temp.getUrlImageLinkedIn());
itemizedOverlay.addOverlay(overlayItem);
```

Fig. 5.18 Posicionament del globus al mapa

5.2.2. iOS

5.2.2.1. Results Direct LinkedIn iPhone

Igual que a la llibreria *LinkedIn-j* utilitzada per Android [66], aquesta llibreria també necessita les claus *consumer key* i un *consumer secret* per poder connectar i utilitzar la API de LinkedIn. La llibreria s'encarrega de facilitar diverses operacions a través de les seves funcions

- La vista per a la identificació d'un usuari.
- L'obtenció dels *Tokens* a través de *Oauth [95]*.
- Establir connexions a la API de LinkedIn a través d'HTTP.

Però a diferència d'altres llibreries aquesta treballa directament amb la notació de la API de LinkedIn, les funcions no venen preestablertes, sinó que existeix una guia de com s'han de crear les funcions. Així doncs, caldrà crear les nostres pròpies funcions. Igual que la llibreria d'Android, només s'utilitzaran els paràmetres necessaris per obtenir les dades, però en aquest cas, aniran directament a la URL [102] de la consulta a la API de LinkedIn.

```
- (RDLinkedInConnectionID *)customProfileForCurrentUser {
    NSURL* url = [NSURL URLWithString: [kAPIBaseURL stringByAppendingString:
        @"/v1/people/~:(id,first-name,last-name,picture-url,positions)"]];
    return [self sendAPIRequestWithURL:url HTTPMethod:@"GET" body:nil];
}
```

Fig. 5.19 Obtenció del perfil de l'usuari

```
- (RDLinkedInConnectionID *)eetacGroup {
    NSURL* url = [NSURL URLWithString: [kAPIBaseURL stringByAppendingString:
        @"/v1/groups/86759:(id,name,short-description,site-group-url,posts:
        (id,creator,creation-timestamp,site-group-post-url,comments,likes,relation-to-
        viewer,attachment:(image-url,content-domain,content-url,title,summary)))"]];
    return [self sendAPIRequestWithURL:url HTTPMethod:@"GET" body:nil];
}
```

Fig. 5.20 Obtenció del grup EETAC Alumni de l'Escola

```
- (RDLinkedInConnectionID *)eetacGroupPopularDiscussions {
    NSURL* url = [NSURL URLWithString: [kAPIBaseURL stringByAppendingString:
        @"/v1/groups/86759:(id,name,short-description,site-group-url,posts:
        (id,creator,creation-timestamp,site-group-post-url,comments,likes,relation-to-
        viewer,attachment:(image-url,content-domain,content-url,title,summary)))
        ?category=discussion&order=popularity"]];
    return [self sendAPIRequestWithURL:url HTTPMethod:@"GET" body:nil];
}
```

Fig. 5.21 Obtenció de les discussions populars del grup

```
- (RDLinkedInConnectionID *)putLikeInEetacGroupWithLike:(NSString*)like
andPostID:(NSString*)postID {
    NSURL* url = [NSURL URLWithString:[kAPIBaseURL stringByAppendingString:[NSString
        stringWithFormat:@"/v1/posts/%@/relation-to-viewer/is-liked", postID]]];
    NSData *likeData = [[NSString stringWithFormat:@"<is-liked>%@</is-liked>", like]
        dataUsingEncoding:NSUTF8StringEncoding];
    return [self sendAPIRequestWithURL:url HTTPMethod:@"PUT" body:likeData];
}
```

Fig. 5.22 Recomanar una discussió del grup

```
- (RDLinkedInConnectionID *)commentPostInEetacGroupWithComment:(NSString*)comment
andPostID:(NSString*)postID {
    NSURL* url = [NSURL URLWithString:[kAPIBaseURL stringByAppendingString:[NSString
stringWithFormat:@"/v1/groups/86759/%@/comments", postID]]];
    NSData *commentData = [comment dataUsingEncoding:NSUTF8StringEncoding];
    return [self sendAPIRequestWithURL:url HTTPMethod:@"POST" body:commentData];
}
```

Fig. 5.23 Comentar una discussió del grup

5.3. Llibreries Internes

A continuació es presentaran les llibreries internes utilitzades pels diferents sistemes operatius (Android i iOS). Per al sistema operatiu Android s'explicarà un altre mètode per guardar les dades. Per iOS s'explicaran les llibreries per persistir les dades (Core Data Framework), la llibreria per accedir a la API de Twitter (Twitter Framework), per serialitzar dades (JSON), per representar mapes (Map Kit Framework) i per utilitzar els serveis de localització (Core Location Framework).

5.3.1. Android

5.3.1.1. Emmagatzematge persistent de dades

A part de la llibreria DB4O [75] per guardar de forma permanent les dades, Android ens proporciona un sistema per emmagatzemar preferències anomenada SharedPreferences. Gràcies a ella, tota la informació que es modifica a la configuració és recordada per l'aplicació un cop es tanca.

Un altre utilitat emprada, ha estat la de guarda variables puntuals, ja que DB4O només permet l'emmagatzematge d'objectes.

Per extreure informació emmagatzemada se li han de passar dos paràmetres, el primer és el nom que té la variable, en aquest cas *allowmessages* i el segon paràmetre és el que es rep per defecte si la variable és buida o no es troba.

```
SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
prefMessages = sharedPrefs.getBoolean("allowmessages", false);
```

Fig. 5.24 Obtenció d'un Booleà de les preferències

5.3.1.2. SendReceived

Aquesta és la classe SendReceived, i és l'encarregada de serialitzar i enviar la informació al servidor i deserialitzar la informació que rep del servidor. Aquesta classe es una llibreria pròpia incorporant les funcions de Gson.

Per facilitar l'entendiment de la classe, la mateixa s'ha separat en dos grups, tot i que al codi original està tota seguida. El primer grup s'encarrega de rebre la informació a serialitzar, la serialitza i li comunica al segon grup a on té que enviar aquesta informació. En el cas de que s'esperi una resposta, el segon grup li passa la informació al primer, aquest la deserialitza i la retorna allà on sigui necessari.

El segon és l'encarregat de realitzar les connexions amb el servidor. Segons el que es necessiti, enviar i rebre dades o només rebre dades, el primer grup escollirà si realitzar un dels dos elements del segon grup, realitzar un POST o realitzar un GET.

5.3.2. iOS

5.3.2.1. Core Data Framework

Core Data és una llibreria que ens facilita Apple i permet guardar objectes de forma persistent. Core Data utilitza com a base de dades SQL Lite, però no s'interacciona directament amb el llenguatge de la base de dades, sinó que es tenen una sèrie d'eines que faciliten tant la creació del model de les dades que es vol guardar, com la seva gestió dins de la base de dades. Core Data es recolza sobre tres conceptes bàsics:

- **Managed Object Data:** és on es defineix el model de les dades.
- **Persistent Storage Coordinator:** és l'encarregat de persistir la informació que es vol guardar.
- **Managed Object Context:** dades temporals on es podrà treballar abans de realitzar la persistència d'aquestes dades.

A l'eina Xcode [103], el fitxer amb l'extensió *.xcdatamodel* és l'encarregat d'emmagatzemar l'estructura del nostre model de dades. A l'obrir aquest fitxer l'Xcode ens ofereix una eina visual que ens permet modelar les dades, creant taules, camps i relacions entre ells.

A l'eina es poden crear diferents *entities* (entitats). I cada entitat contindrà l'estructura de dades que es vol guardar. Per a l'aplicació s'ha creat l'entitat *Me*. Una entitat disposa de *properties* (propietats), aquestes propietats poden ser del tipus *atributte* (atribut o camp) o reltaionship (relacions). Els camps de l'estructura són molt semblants als utilitzats a l'objecte *EetacPerson*. Les relacions serveixen per unir diferents entitats, en el nostre cas no s'han utilitzat.

Per accedir i llegir les dades persistents cal crear un objecte *NSFetchRequest* per realitzar la consulta. A continuació s'ha de definir la entitat on es vol realitzar la consulta i a continuació ja es pot realitzar la consulta. Per últim es guarda en una variable les dades obtingudes, comprovant si hi ha hagut algun error.

```
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

NSEntityDescription *entity = [NSEntityDescription entityForName:@"Me"
inManagedObjectContext:managedObjectContext];
[fetchRequest setEntity:entity];

NSError *error = nil;
NSManagedObject *meObject = [[managedObjectContext executeFetchRequest:fetchRequest
error:&error] lastObject];
EetacPerson *me = meObject;
```

Fig. 5.25 Llegir informació procedent del model de dades

Per inserir informació a les dades persistents, primer es crea l'objecte que es vol persistir indicant a quina entitat pertany. A continuació ja es pot omplir l'objecte amb la informació adequada. I per últim ja es pot guardar l'objecte comprovant que no hi hagi cap error.

```
Me *coreDataEetacPerson = (Me *)[NSEntityDescription insertNewObjectForEntityForName:@"Me"
inManagedObjectContext:managedObjectContext];

[coreDataEetacPerson setStudent:me.student];
[coreDataEetacPerson setEmail:me.email];
[coreDataEetacPerson setAllowEmail:[NSNumber numberWithBool:me.allowEmail]];
…
NSError *error;
[managedObjectContext save:&error];
```

Fig. 5.26 Inserir informació al model de dades

Per actualitzar informació a les dades persistents, es realitza la mateixa operació que s'ha fet per llegir un objecte, però afegint més operacions. Després de llegir l'objecte es modificaran els valors dels camps desitjats i seguidament es tornarà a guardar l'objecte modificat.

```
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Me"
inManagedObjectContext:managedObjectContext];
    [fetchRequest setEntity:entity];

NSError *error = nil;
    EetacPerson *coreDataMe = [[managedObjectContext executeFetchRequest:fetchRequest
error:&error] lastObject];

coreDataMe.student = me.student;
            coreDataMe.email = me.email;
            [coreDataMe setValue:[NSNumber numberWithBool:me.allowEmail]
forKey:@"allowEmail"];

[managedObjectContext save:&error]
```

Fig. 5.27 Actualitzar informació al model de dades

Per eliminar informació a les dades persistents, només fa falta identificar quin objecte es vol eliminar i a continuació eliminar-lo.

```
NSManagedObject *meObject = [[managedObjectContext executeFetchRequest:fetchRequest
error:&error] lastObject];
[managedObjectContext deleteObject:meObject];
```

Fig. 5.28 Eliminar informació al model de dades

5.3.2.2. Map Kit i Core Location Framework

A l'aplicació, aquestes dues llibreries ens donen els objectes, funcions i protocols necessàries per crear mapes, definir xinxetes o pins i establir la nostre posició a través dels serveis de localització del telèfon mòbil, ja sigui a través de la xarxa de telefonia o a través del sistema de posicionament global (GPS).

Per crear un mapa cal afegir un *MapView* a la interfície gràfica i enllaçar-la amb el codi a través d'un *IBOutlets*. Un cop creat el mapa, quan s'interacciona amb ell, la llibreria disposa de diferents funcions per cada acció realitzada.

```
- (void)mapView:(MKMapView *)mapView regionDidChangeAnimated:(BOOL)animated {}
```

Fig. 5.29 Funció que s'executa quan es mou, s'amplia o es redueix el mapa

Per crear xinxetes al mapa primer de tot s'ha de definir el pin utilitzant l'objecte *MKAnnotation*, amb aquest objecte es podrà personalitzar la localització, l'aspecte i afegir certa informació a la xinxeta, després només serà necessari afegir aquest objecte al mapa.

Quan es prem un pin, aquest disposa d'una petita vista on es mostra la informació sobre aquest pin i a través d'un botó anomenat accessori es accedir a altres vistes amb més informació.

```
MKAnnotationView *annotation = [[MKAnnotationView alloc] init];
[annotation setTitle:eetacPerson.name];
[annotation setImage:[UIImage imageWithData:imageData]];
MKCoordinateRegion region = { {0.0, 0.0 }, { 0.0, 0.0 } };
region.center.latitude = eetacPerson.lat;
region.center.longitude = eetacPerson.lon;
[annotation setCoordinate:region.center];
[mapView addAnnotation: annotation];
```

Fig. 5.30 Exemple de creació d'una xinxeta

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView
            viewForAnnotation:(id <MKAnnotation>)annotation {}
```

Fig. 5.31 Funció que s'executa al prémer una xinxeta

```
- (void)mapView:(MKMapView *)mapView annotationView:(MKAnnotationView *)view
                      calloutAccessoryControlTapped:(UIControl *)control {}
```

Fig. 5.32 Funció que s'executa al prémer l'accessori de la xinxeta

Per trobar la nostra localització s'hauran d'iniciar els serveis de localització de l'iPhone. Es podrà definir a partir de quin grau de precisió es rebran les dades de localització del servei. Un cop rebudes les dades només s'haurà d'afegir la nostra localització al mapa. Aquesta localització és tracta igual que una xinxeta.

```
locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.distanceFilter = kCLDistanceFilterNone;
locationManager.desiredAccuracy = kCLLocationAccuracyKilometer;

[locationManager startUpdatingLocation];

[mapView setShowsUserLocation:YES];
```

Fig. 5.33 Posar en marxa el servei de localització

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation {}
```

Fig. 5.34 Funció on es reben les dades de localització

5.3.2.3. Accounts and Twitter Framework

Aquestes dues llibreries forment part del sistema operatiu iOS i faciliten la connexió amb la API [68] de Twitter. En aquest cas, la llibreria Accounts és l'encarregada d'accedir a la compte de Twitter prèviament configurada a la Configuració de iOS. El sistema operatiu s'encarrega de gestionar l'autenticació amb l'API de Twitter a través d'*Oauth [95]*, de manera que no és necessari obtenir una les claus *consumer key* i *consumer secret* per a l'aplicació "Sóc de l'EETAC". La llibreria Twitter és l'encarregada de mostrar la vista per escriure nous tuits i de realitzar les peticions amb autenticació a la API de Twitter.

Per obtenir o rebre les dades necessàries per l'aplicació, es realitzaran consultes directament a la API de Twitter, igual que a Android serà necessari recuperar els tuits i mencions i obtenir l'estat del nostre usuari i obtenir també el *User Timeline* de l'usuari de l'Escola (@EETAC_UPC). També serà necessari reenviar tuits (repiular [99]).

```
ACAccountStore *account = [[ACAccountStore alloc] init];
ACAccountType *accountType = [account accountTypeWithAccountTypeIdentifier:
      ACAccountTypeIdentifierTwitter];

[account requestAccessToAccountsWithType:accountType withCompletionHandler:^(BOOL granted,
NSError *error) {
        if (granted == YES) {
            NSArray *arrayOfAccounts = [account accountsWithAccountType:accountType];
            if (arrayOfAccounts > 0) {
                ACAccount *twitterAccount = [arrayOfAccounts objectAtIndex:0];
…
```

Fig. 5.35 Accedir a la compte de Twitter del sistema

```
- (void)fetchTweets {
    NSMutableDictionary *params = [[NSMutableDictionary alloc] init];
    [params setObject:me.twitterName forKey:@"screen_name"];
    [params setObject:@"10" forKey:@"count"];
    [params setObject:@"1" forKey:@"include_entities"];
    [params setObject:@"1" forKey:@"include_rts"];

    NSURL *urlUser = [NSURL URLWithString:
        @"https://api.twitter.com/1/statuses/user_timeline.json"];
    NSURL *urlMentions = [NSURL URLWithString:
        @"https://api.twitter.com/1/statuses/mentions.json"];

    TWRequest *requestUser = [[TWRequest alloc] initWithURL:urlUser parameters:params
requestMethod:TWRequestMethodGET];
    TWRequest *requestMentions = [[TWRequest alloc] initWithURL:urlMentions parameters:params
requestMethod:TWRequestMethodGET];

    [requestUser performRequestWithHandler:^(NSData *responseData, NSHTTPURLResponse
*urlResponse, NSError *error) {
…
    [requestMentions performRequestWithHandler:^(NSData *responseData, NSHTTPURLResponse
*urlResponse, NSError *error) {
…
```

Fig. 5.36 Obtenir tuits i mencions

```
- (IBAction)tweetCompose:(id)sender {
    BOOL canSendTweet = [TWTweetComposeViewController canSendTweet];
    if (canSendTweet) {
        TWTweetComposeViewController *twitter = [[TWTweetComposeViewController alloc] init];
        [twitter setInitialText:@"#socdelEETAC "];

        [self presentModalViewController:twitter animated:YES];

        twitter.completionHandler = ^(TWTweetComposeViewControllerResult result) {
            …
            [self dismissModalViewControllerAnimated:YES];
        };
    }
}
```

Fig. 5.37 Enviar un tuit

```
- (IBAction)retweetMessage:(id)sender {
    NSString *retweetString = [NSString stringWithFormat:
        @"http://api.twitter.com/1/statuses/retweet/%@.json", myTwitterName];
    NSURL *retweetURL = [NSURL URLWithString:retweetString];
    TWRequest *request = [[TWRequest alloc] initWithURL:retweetURL parameters:nil
        requestMethod:TWRequestMethodPOST];
    [request performRequestWithHandler:^(NSData *responseData, NSHTTPURLResponse
        *urlResponse, NSError *error) {
        …
```

Fig. 5.38 Repiular

5.3.2.4. JSON

Apple dóna suport natiu per JSON [92], permetent la conversió d'objectes a aquest format a i l'inrevés. Aquesta llibreria és una llibreria pròpia utilitzant les diferents funcions disponibles de JSON i integrant també les diferents funcions disponibles per realitzar connexions als servei web. El resultat és una llibreria que s'encarrega de serialitzar i deserialitzar objectes i establir connexions, enviar i rebre informació a través d'internet utilitzant els mètodes del servei web.

Cada classe de cada vista que necessiti utilitzar aquesta llibreria bàsicament farà servir tres tipus de funcions, la primera realitzarà una petició per obtenir dades del servidor, la segona realitzarà una petició enviant primer dades cap al servidor i rebent-ne unes altres, i per últim una altra on es recolliran les dades rebudes del servidor.

A la primera funció utilitzarem el mètode GET [79] del servei web per rebre les dades del servidor, aquesta connexió es realitzarà de forma asíncrona, permetent a l'aplicació seguir amb la seva execució sense haver d'esperar a la recepció de les dades. A aquesta funció se li passarà un paràmetre *path* que és el camí parcial a la URL [102] (adreça web) on la llibreria realitzarà la connexió per obtenir les dades. La funció muntarà la URL sencera amb el paràmetre *path*, la part fixe de l'adreça i la IP del servidor. Després realitzarà la connexió establint un temps d'espera de 10 segons en cas de no rebre resposta del servidor.

```
- (void)getAsyncronousJsonWithPath:(NSString*)path {

    urlDelegate = self;
    NSString *serverIP = [IP getServerIP];
    NSURL *url = [NSURL URLWithString:[NSString stringWithFormat:
        @"http://%@:8080/ImFromEetac/rest/ImFromEetac/%@", serverIP, path]];
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
        cachePolicy:NSURLRequestUseProtocolCachePolicy timeoutInterval:10.0];
    id urlConnection = [[NSURLConnection alloc] initWithRequest:request
        delegate:urlDelegate];
}
```

Fig. 5.39 Primera funció on es realitza la petició de dades (mètode GET)

Les dades de la connexió es reben prèviament a una funció de la llibreria i es deserialitzaran i es guardaran en una variable genèrica abans d'enviar-les a la tercera funció que està implementada a la vista d'origen.

```
- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    if (asyncronousJSONResponse != nil) {
        NSError *error = nil;
        id result = [NSJSONSerialization JSONObjectWithData:asyncronousJSONResponse
            options:NSJSONReadingMutableContainers error:&error];
        NSLog(@"%@", result);
            [jsonDelegate receivedJSONDictionary:(NSDictionary*)result];
```

```
        }
        else {
            NSLog(@"JSON error: %@", error);
        }
    }
    else {
        NSLog(@"JSON data not received!");
    }
}
```

Fig. 5.40 Funció on es reben i deserialitzen les dades rebudes del servidor

A la segona funció utilitzarem el mètode POST [97] del servei web per rebre les dades del servidor, aquesta connexió també es realitzarà de forma asíncrona, permetent a l'aplicació seguir amb la seva execució sense haver d'esperar a la recepció de les dades. A aquesta funció se li passarà un paràmetre *path* que és el camí parcial a la URL (adreça web) on la llibreria realitzarà la connexió per obtenir les dades. A diferència de la primera funció, en aquesta, se li passarà un segon paràmetre amb la informació que es vol enviar al servidor a través d'un objecte genèric. Primer de tot, la funció comprovarà si l'objecte que se li ha passat es pot convertir al format JSON [92], i si és així farà la conversió. Igual que la primera funció, aquesta també muntarà la URL sencera amb el paràmetre *path*, la part fixe de l'adreça i la IP del servidor. Per últim, es muntarà la petició afegint la informació ja convertida en format JSON a utilitzant el mètode POST i seguidament es realitzarà la connexió establint un temps d'espera de 10 segons en cas de no rebre resposta del servidor.

```
- (void)postAsyncronousJsonFromNSDictionary:(NSDictionary*)dataDictionary
withPath:(NSString*)path {
    urlDelegate = self;
    if ([NSJSONSerialization isValidJSONObject:dataDictionary]) {
        NSError *error = nil;
        NSData *postDictionaryData = [NSJSONSerialization dataWithJSONObject:dataDictionary
            options:NSJSONWritingPrettyPrinted error:&error];
        if (error == nil && postDictionaryData != nil) {
            NSString *serverIP = [IP getServerIP];
            NSURL *url = [NSURL URLWithString:[NSString stringWithFormat:
                @"http://%@:8080/ImFromEetac/rest/ImFromEetac/%@", serverIP, path]];
            NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
                cachePolicy:NSURLRequestUseProtocolCachePolicy timeoutInterval:10.0];
            [request setHTTPMethod:@"POST"];
            [request setValue:@"application/json" forHTTPHeaderField:@"Accept"];
            [request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
            [request setValue:[NSString stringWithFormat:@"%d", [postDictionaryData length]]
                forHTTPHeaderField:@"Content-Length"];
            [request setHTTPBody:postDictionaryData];
            [request setTimeoutInterval:10.0];
            id urlConnection = [[NSURLConnection alloc] initWithRequest:request
                delegate:urlDelegate];
        }
        else {
            NSLog(@"%@\n", [error localizedDescription]);
        }
    }
}
```

Fig. 5.41 Funció on es realitza la petició de dades a través del mètode POST

Per últim, a la tercera funció, és l'encarregada de rebre les dades ja deserialitzades a la classe que havia fet la petició a través de la primera o segona funció descrites anteriorment. Cada classe serà l'encarregada d'interpretar i convertir les dades rebudes contingudes a l'objecte genèric de la funció al corresponent objecte utilitzat en la classe.

```
- (void)postAsyncronousJsonFromNSDictionary:(NSDictionary*)dataDictionary
withPath:(NSString*)path {
    urlDelegate = self;
    if ([NSJSONSerialization isValidJSONObject:dataDictionary]) {
        NSError *error = nil;
        NSData *postDictionaryData = [NSJSONSerialization dataWithJSONObject:dataDictionary
            options:NSJSONWritingPrettyPrinted error:&error];
        if (error == nil && postDictionaryData != nil) {
            NSString *serverIP = [IP getServerIP];
            NSURL *url = [NSURL URLWithString:[NSString stringWithFormat:
                @"http://%@:8080/ImFromEetac/rest/ImFromEetac/%@", serverIP, path]];
            NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
                cachePolicy:NSURLRequestUseProtocolCachePolicy timeoutInterval:10.0];
            [request setHTTPMethod:@"POST"];
            [request setValue:@"application/json" forHTTPHeaderField:@"Accept"];
            [request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
            [request setValue:[NSString stringWithFormat:@"%d", [postDictionaryData length]]
                forHTTPHeaderField:@"Content-Length"];
            [request setHTTPBody:postDictionaryData];
            [request setTimeoutInterval:10.0];
            id urlConnection = [[NSURLConnection alloc] initWithRequest:request
                delegate:urlDelegate];
        }
        else {
            NSLog(@"%@\n", [error localizedDescription]);
        }
    }
}
```

Fig. 5.42 Tercera funció on es realitza la petició de dades (mètode POST)

5.4. Implementació i creació de vistes

En aquest apartat es veurà com es genera i s'implementa una vista amb els diferents sistemes operatius de l'aplicació (Android i iOS).

5.4.1. Android

Tenim dues opcions per construir una interfície a Android, definint-la mitjançant codi a la pròpia activitat o generant-la en un fitxer *XML* i accedint-hi a través del mètode *onCreate()*.

També es pot realitzar una combinació de les dues, creant certs elements que seran fixos en el fitxer *XML* i afegint altres mitjançant codi quan sigui necessari. Com a exemple tenim la construcció de taules: es defineix la taula i els seus atributs a *l'XML*, i es s'afegeixen files conforme es vagin llegint més dades.

Es mostrarà un exemple d'aquest cas: quan es prem el globus al mapa a la informació d'usuari. Amb aquest exemple es veuran els dos casos: la creació d'un element visual mitjançant *XML* i el seu accés al codi; i també la creació d'elements visuals des de codi i la seva forma d'afegir-los per mostrar-los per pantalla. Al fitxer *linkedin.xml*, en el seu interior es defineixen, entre altres coses, una *TableLayout* amb les següents característiques:

```
<TableLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/jobstable"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="0,1"
        android:background="@drawable/border">
</TableLayout>
```

Fig. 5.43 Creació d'una taula des de l'arxiu XML

A la **figura 5.43** es mostra el codi que defineix la taula per mostrar la informació extra d'un titulat, concretament, l'apartat de Feina actual de LinkedIn, ja que una persona pot ser que tingui més d'una feina, cas poc habitual, però possible. Per aquest motiu s'ha de fer una taula dinàmica, que mostri la informació correctament.

Fig. 5.44 Detall de la feina actual d'un titulat

A l'esquerra de la **figura 5.44** s'observa el títol, i a la dreta el contingut. Per modificar les taules al codi, el que s'ha de fer, és accedir-hi mitjançant el seu identificador a la funció *onCreate()*.

```
country_table=(TableLayout)findViewById(R.id.jobstable);
```

Fig. 5.45 Accés a la taula definida a *l'XML* a través del seu identificador

Un cop es té l'objecte de tipus *TableLayout* instanciat, per afegir files n'hi ha prou utilitzant un bucle, on a cada iteració es llegeixi la informació referent a la feina i s'afegeix una fila amb la informació al final de cada iteració. A continuació es mostra un resum del codi que afegeix files a la taula, on t1 i t2 són variables amb la informació. Les variables senars mostren els títols i les variables parells mostren la descripció del contingut.

```
t1.setText("des de ");
t1.setTextColor(Color.parseColor("#00a4f9"));
t1.setGravity(Gravity.RIGHT);
t1.setLayoutParams(new
TableRow.LayoutParams(0,android.view.ViewGroup.LayoutParams.WRAP_CONTENT,3));

t2.setText(temp.getStartJobDate());
t2.setTextColor(Color.parseColor("#404040"));
t2.setLayoutParams(new
TableRow.LayoutParams(0,android.view.ViewGroup.LayoutParams.WRAP_CONTENT,10));
```

Fig. 5.46 Definició dels *TextViews* t1 i t2 de forma dinàmica

```
for(Job temp:CurrentJob) {
        row = new TableRow(this);
        row2 = new TableRow(this);
        row3 = new TableRow(this);
…
        row.addView(t1);
        row.addView(t2);
        row2.addView(t3);
        row2.addView(t4);
        row3.addView(t5);
        row3.addView(t6);
…
        country_table.addView(row, new TableLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        country_table.addView(row2, new TableLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        country_table.addView(row3, new TableLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
}
```

Fig. 5.47 Accés a la taula definida a *l'XML* a través del seu identificador

Per a una major claredat del codi, s'han omès altres sentències que apareixen en el flux de dades implementat, però en essència aquest és el model a seguir per interactuar entre dades estàtiques i dinàmiques d'interfície. A cada iteració, s'instancia de nou cada element que es vulgui afegir i se li dóna valor als atributs que es necessiten.

Un element més de la part visual d'Android, que s'ha utilitzat per fer l'aplicació, és el *ListView*. Un *ListView* és un sistema per mostrar una gran quantitat d'elements en forma de llista, com ara els missatges de Twitter, correu electrònics, estadístiques, etc. Els seus principals avantatges són: és molt còmode d'usar i és molt eficient.

Aquest element, per defecte, és visualment una mica senzill, ja que només permet escriure text, i no deixa col·locar imatges. Com s'observa a continuació, el resultat millora l'aspecte visual de l'aplicació, ja que s'han afegit un parell d'imatges, la de la web i la de la fletxeta a través de *l'Adapter*.

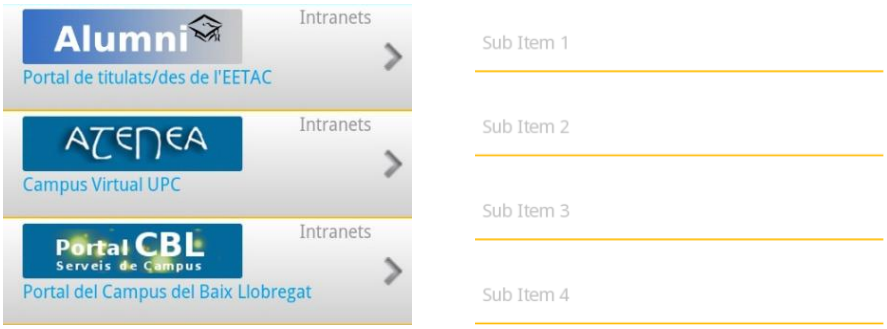

Fig. 5.48 *ListView* amb i sense *Adapter* respectivament

Per omplir un *ListView* s'ha de passar un element de tipus *Adapter* juntament amb les dades es vol que contingui la llista. Normalment per a un *ListView* simple, s'utilitza la classe *ArrayAdapter*, però com el nostre *ListView* serà personalitzat, s'ha de crear el nostre propi *Adapter*. Per a realitzar això, simplement s'ha de crear una classe que hereti de *BaseAdapter* i sobreescriure els mètodes que conté aquesta classe abstracta.

El primer que s'ha de fer per crear una llista on poder inserir imatges, és crear un nou *layout* [93] amb l'aspecte que es vol aconseguir per cada element. També s'ha de crear un nou arxiu per aconseguir l'efecte de degradat, que proporciona un efecte de volum al llistat.

Ara ja es té definit l'aspecte de cada element de la llista. A continuació s'ha de crear un altre *Layout* on estigui la llista en si, sense res més, l'aspecte seria com el mostrat anteriorment a la **figura 5.48 (*ListView* sense *Adapter*)**.

La part visual ja estaria complerta, ara només faltaria emplenar la llista amb els elements desitjats.

Una vista que difereix de les demés, és la primera de totes, la que apareix al executar l'aplicació per primera vegada, la d'*Accés*. Els mecanismes per crear aquesta vista són els mateixos que la resta, mitjançant l'editor que ens proporciona el SDK de Google, però per aconseguir l'efecte de finestra flotant cal afegir una línia al *AndroidManifest* fent referencia al canvi de tema.

```
android:theme="@android:style/Theme.Dialog"
```

Fig. 5.49 Canviant el tema des de l'*AndroidManifest*

Per el cas de la pantalla de l'apartat Opcions, Android [66] ofereix una forma de construir aquesta finestra de manera automàtica i mostrant una aparença similar a la de totes les finestres de configuració del sistema operatiu.

Per a la creació de la vista de configuració seran necessaris dos arxius, un arxiu *XML* anomenat *preferences.xml,* on es definiran els elements que es visualitzaran, i una nova *Activity* [65] que heretarà de *PreferenceActivity* anomenada *Preferences.java*.

```
<PreferenceCategory android:title="@string/config_social_title" >
        <CheckBoxPreference
                        android:defaultValue="false"
                        android:key="twitter"
                        android:summary="@string/config_twittervinc_sum"
                        android:title="@string/config_twittervinc" >
        </CheckBoxPreference>
        <CheckBoxPreference
                android:defaultValue="false"
                android:key="linkedin"
        android:summary="@string/config_linkedinvinc_sum"
        android:title="@string/config_linkedinvinc" >
        </CheckBoxPreference>
</PreferenceCategory>
```

Fig. 5.50 Detall del arxiu *preferences.xml*

Fig. 5.51 Resultat visual del arxiu *preferences.xml*

5.4.2. iOS

L'Xcode [103] disposa d'una eina anomenada *Interface Builder* que facilita la creació de vistes d'una forma fàcil i visual, només cal arrastrar diferents objectes a cada vista i col·locar-los allà on ens faci falta. Una vista també es pot realitzar tota a través de codi. Per a l'aplicació s'han utilitzat els dos mètodes conjuntament, aprofitant la senzillesa de crear vistes visualment i aprofitant la versatilitat del codi per muntar vistes dinàmiques.

5.4.2.1. Interface Builder

Quan accedim a l'eina *Interface Builder* es mostra una finestra amb un espai de treball anomenat *Storyboard*, en aquest espai es poden anar afegint vistes i enllaçar-les amb d'altres, de manera que al final queda un organigrama amb totes les vistes de l'aplicació. L'eina també disposa d'un llistat a la part esquerra de totes les vistes i elements que una vista té agregades i un apartat a la dreta ple d'opcions per a cada vista i objecte que es seleccioni.

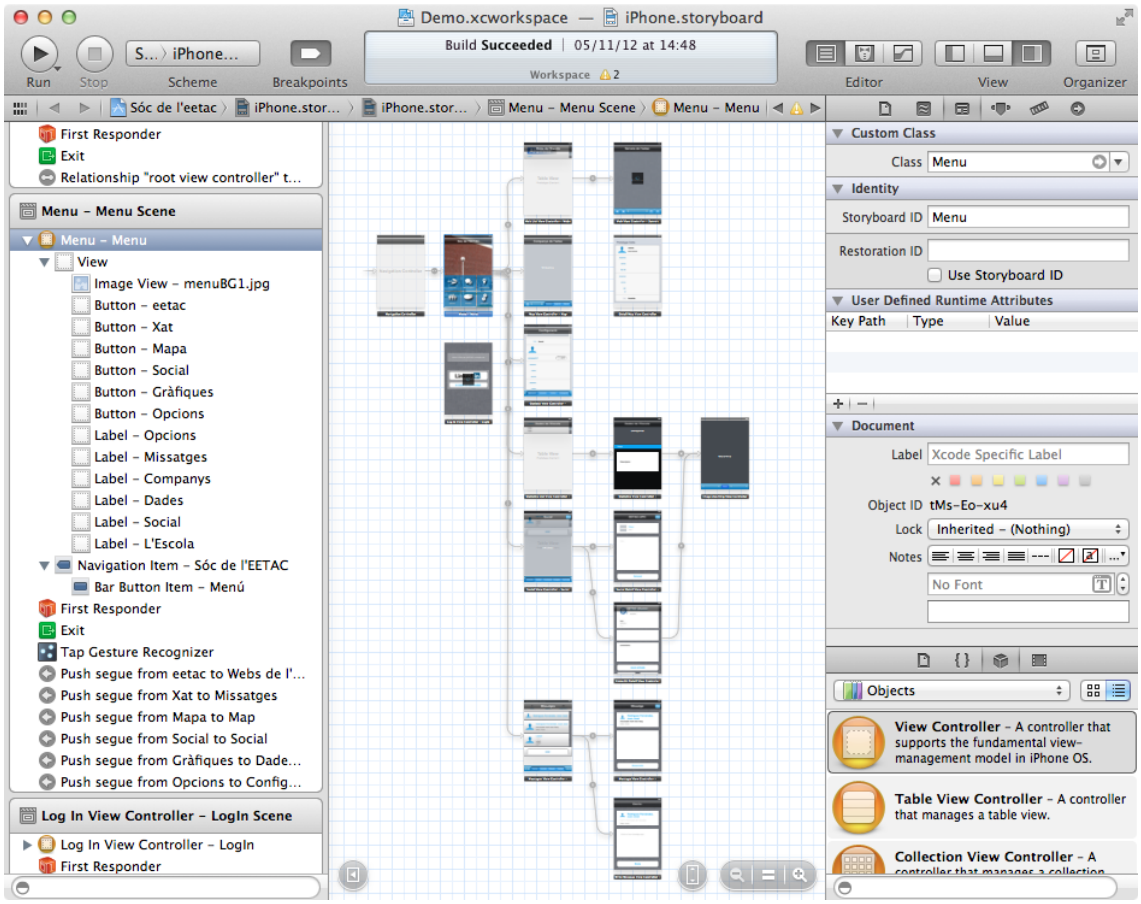

Fig. 5.52 *Interface Builder* i *Storyboard*

Per afegir vistes a l'*Storyboard* només fa falta arrastrar el tipus de vista que es vol des de l'apartat d'opcions. Un cop es té la vista a l'espai de treball, per afegir diferents elements a la vista com ara un botó un a caixa de text, un llistat... es realitza de la mateixa forma, arrastrant l'element seleccionat a la vista i col·locant-lo on ens convé, cada element que s'afegeix es pot redimensionar visualment de forma ràpida a través del seu contorn.

Però això no es tot, per crear la interfície gràfica de l'aplicació, cada vista que es creï s'ha d'enllaçar en amb una classe anomenada *ViewController* que controli aquesta vista. Cada element que s'afegeixi a la vista si és necessari pot ser enllaçat amb aquesta classe a través d'unes propietats anomenades

IBOutlets. També hi ha unes funcions anomenades *IBActions*, que són les encarregades d'enllaçar una acció, per exemple quan es prem un botó.

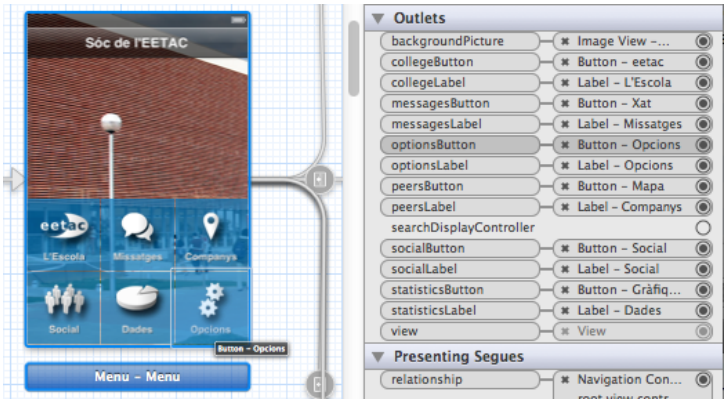

Fig. 5.53 Enllaços (IBOutlets) de la vista Menú

```
@property (nonatomic, strong) IBOutlet UIButton *optionsButton;
- (IBAction)goToOptions:(id)sender;
```

Fig. 5.54 Codi de l'*IBOutlet* i *IBAction* del botó Opcions

Al declarar un *IBOutlet* i enllaçar-lo, a través de codi es pot accedir a totes les opcions de l'element enllaçat. Per tant es pot utilitzar senzillesa de muntar la interfície gràfica visualment i aprofitar totes les opcions de que disposa el codi per l'element enllaçat.

Per presentar i enllaçar varies vistes hi ha diverses tècniques, a l'aplicació s'ha utilitzat el sistema de navegació (*Navigation Controller*). Aquest sistema és una plantilla auto programada per facilitar el desplaçament entre diverses vistes, concretament porta integrada una barra a la part superior on podem indicar el títol de la vista i en aquesta barra apareixen automàticament els botons per tornar la vista anterior un cop s'ha enllaçat dues vistes a través d'un element anomenat Segues. Els Segues són les línies que apareixen entre les vistes a la **figura 5.52**, se'ls hi pot assignar un identificador i un efecte de transició entre vista i vista. Cada cop que s'enllaça una vista, aquesta entra a formar part del sistema de navegació i no s'ha d'escriure cap mena de codi perquè el sistema funcioni. Per activar un canvi de vista a través d'un Segue, és pot fer directament a través de les accions d'un botó, en compte d'assignar una *IBAction* al codi, es pot assignar una acció a *Triggered Segues*; o a través del codi. Si es fa a través de codi, cal identificar a través de quin *Segue* es vol fer el canvi de vista i també es pot aprofitar per traspassar variables a la nova vista.

5.4.2.2. Amb codi

A través de codi també es pot muntar la interfície gràfica, té l'avantatge de poder inserir nous elements a la vista o eliminar-los, és adient per elements dinàmics. A través del codi també es poden definir més opcions d'un element que a través de l'*Interface Builder*. La desavantatge és que cal escriure més línies de codi que si es fa visualment. Hi haurà casos en que només es podrà utilitzar el codi per afegir elements a una vista.

```
UIButton *loginButton = [[UIButton alloc] init];

loginButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[loginButton setTitle:@"Connecta" forState:UIControlStateNormal];
[loginButton setTitleColor:[UIColor colorWithRed:0.0 green:0.47843 blue:0.77647 alpha:1.0]
      forState:UIControlStateNormal];
 [loginButton setTintColor:[UIColor colorWithRed:0.0 green:0.47843 blue:0.77647 alpha:1.0]];
[loginButton setFrame:CGRectMake(10.0, 16.0, 225.0, 44.0)];
loginButton.enabled = YES;

[loginButton addTarget:self action:@selector(login:)
      forControlEvents:UIControlEventTouchUpInside];

[footerView addSubview:loginButton];
```

Fig. 5.55 Exemple de codi d'un botó

A l'exemple es pot veure com primer es el botó. A continuació s'assignen les propietats i les diferents opcions per al botó, com ara el nom que apareix dins del botó, el color del text, el color de fons quan es prem el botó. A continuació també li assignem una acció (*IBAction*). I per últim s'afegeix el botó a una altre vista ja existent.

Alguns elements, a part de propietats i opcions, disposen de varies funcions ja que aquests elements visuals són interactius, un exemple són les taules (*UITableView*). Aquest element disposa de funcions per controlar quantes seccions pot tenir una taula, per indicar quantes cel·les pot tenir cada secció, per indicar la mida de les cel·les, per indicar quina acció és realitza al prémer una cel·la, etc.
Per últim, una part no menys important de la interfície gràfica, resideix en unes funcions, que cada classe ViewController té a la seva disposició.

- ***viewDidLoad*:** aquesta funció s'executa quan es carrega per primera vegada la vista. Pot servir per activar diferents elements gràfics.
- ***viewDidUnload*:** aquesta funció s'executa quan s'abandona una vista. Bàsicament servei per alliberar memòria del elements de cada vista.
- ***viewWillAppear*:** aquesta funció s'executa quan la vista apareix, ja sigui per primera o més vegades. S'executa després de *viewDidLoad*.

- ***viewWillDisappear:*** aquesta funció s'executa quan estem a punt de canviar de vista. Pot servir per guardar dades al tancar la vista.
- ***shouldAutorotateToInterfaceOrientation:*** aquesta funció permet definir si la vista pot rotar cap als modes apaïsats o verticals.
- ***willRotateToInterfaceOrientation:*** aquesta funció permet definir quines accions s'han d'executar quan es rotin les vista. Per exemple moure de lloc algun element gràfic.

5.4.2.3. Quartz Core i Core Graphics

Aquestes dues llibreries són les responsables d'afegir retocs gràfics als elements de les vistes i de permetre animacions d'aquests elements. Per exemple arrodonir cantonades, afegir un marc d'un gruix i color determinat o afegir una ombra a un element. En el cas de les animacions per exemple es pot fer aparèixer o desaparèixer un element de la vista utilitzant una transició de dissolució, volteig... assignant varies opcions com la durada de la transició.

```
self.messageView.layer.cornerRadius = 8.0;
self.messageView.layer.borderColor = [[UIColor grayColor] CGColor];
self.messageView.layer.borderWidth = 1;
self.messageView.layer.masksToBounds = YES;
```

Fig. 5.56 Exemple per arrodonir cantonades i afegir un marc a un element

```
[UIView animateWithDuration:0.5 delay:0.0 options:UIViewAnimationCurveEaseOut animations:^{
    alertView.hidden = NO;
    center = alertView.center;
    center.y -= 44;
    alertView.center = center;
}
completion:^(BOOL finished) {
…
```

Fig. 5.57 Animació, aparèixer i moure una vista (missatge d'alerta)

5.5. Implementació del mòdul d'Accés i Menú

En aquest apartat s'explicarà el funcionament del mòdul d'accés i el menú de l'aplicació, i també es veurà la implementació d'aquest mòdul segons el sistema operatiu emprat (Android i iOS).

5.5.1. Funcionament

A continuació es mostra el diagrama amb el funcionament bàsic de les vistes d'Accés i Menú de l'aplicació i la seva interacció amb el servidor "Sóc de l'EETAC" i l'API [68] de LinkedIn.

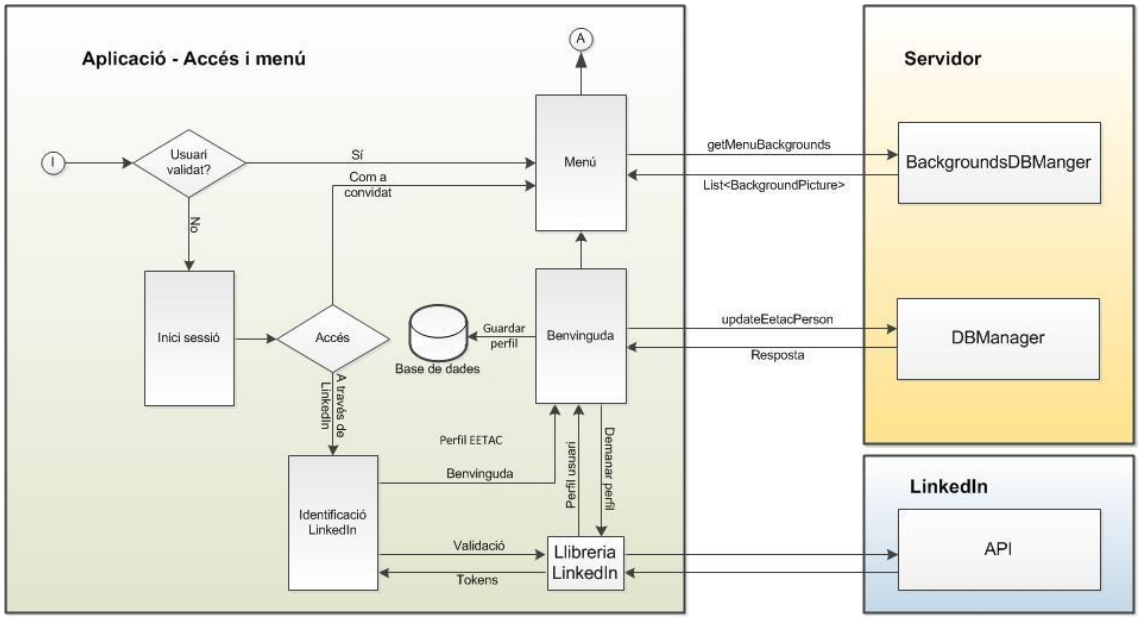

Fig. 5.58 Diagrama de funcionament de l'Accés i Menú a l'aplicació

Quan s'executa l'aplicació encara sigui per primera vegada es comprova si l'usuari s'ha validat a través de la xarxa professional de LinkedIn. Aquesta validació és realitza accedint a la base de dades interna i comprovant el valor del camp *allowAcces*. Si l'usuari no està validat automàticament es carrega la vista *Inici sessió* on es pot triar el mode d'accés a l'aplicació. Es pot accedir a l'aplicació com a convidat, i llavors s'accediria directament al *Menú*. Si accedim a l'aplicació a través de LinkedIn ens apareixerà una nova vista (*Identificació LinkedIn*) que permetrà validar-se a la xarxa professional LinkedIn. Un cop validats rebrem una missatge de benvinguda i s'accedirà automàticament al menú de l'aplicació. Des del menú es podrà accedir als diferents apartats de que disposa l'aplicació: L'Escola, Missatges, Companys, Social, Dades i Opcions.

5.5.2. Android

A continuació es presentaran les implementacions de la interfície gràfica Inici sessió, Identificació LinkedIn i Menú, presentats anteriorment a la **figura 5.58**. Vegeu la **Fig. I.2** Accés a l'aplicació, Android i iOS respectivament, la **Fig. I.3** Autentificació a LinkedIn, Android i iOS respectivament i la **Fig. I.4** Vista del

Menú, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.5.2.1. Inici sessió

Per accedir a l'aplicació no nomes cal estar validat al LinkedIn, també cal que la persona estigui a la base de dades del servidor.

Un cop hem rebut les claus que identifiquen el usuari es realitza la primera connexió al LinkedIn demanant informació de la persona. Tot seguit el que realitza l'aplicació és enviar als nostres servidors el nom el cognom i el identificador del LinkedIn, compara el nom i cognom i si hi ha coincidència retorna un objecte del tipus EetacPerson amb informació del usuari.

```
Send senddata = new Send();
NewID newid = new NewID();
newid.setName(personli.getFirstName());
newid.setSurname(personli.getLastName());
newid.setLinkedInID(personli.getId());
person = senddata.postData(newid);
```

Fig. 5.59 Enviant usuari al servidor de l'Escola per primera vegada

Tot seguit ja es te al usuari validat en el LinkedIn i en el servidor, ara cal actualitzar les dades i el que es far és tornar a connectar amb el servidor per enviar la resta d'informació que es te del usuari, que és bàsicament el llistat de feines passades com actuals i la direcció web de la imatge del LinkedIn. Més endavant l'usuari podrà decidir si vol o no compartir aquesta informació.

5.5.2.2. Identificació LinkedIn

La validació amb LinkedIn es realitza gràcies a l'ajut de les llibreries LinkedIn-j i Signpost.

Un cop es prem el botó de LinkedIn l'aplicació envia les claus úniques que identifiquen l'aplicació als servidors de LinkedIn i tot seguit s'obre el navegador on l'usuari introdueix les seves credencials. Tot seguit l'aplicació rep les claus que identifiquen al usuari, hi ha continuació es guarden de forma persistent per futures connexions.

5.5.2.3. Menú

El menú principal de l'aplicació esta composat de sis botons, que donen accés als diferents apartats de l'aplicació.

Per realitzar el canvi de vista el sistema Android utilitza el que anomena *Intent.* Un Intent és una crida a una altra aplicació, ja sigui de la nostra aplicació o del sistema operatiu Android. Dins d'aquesta crida podem afegir dades, ja siguin paràmetres de configuració o simple informació.

Primer cal crear una variable de tipus *Button* i vincular-la al boto al boto creat al arxiu *XML*. Un cop fet, es crea un mètode que li digui a l'aplicació que cal fer quant es prem el boto, i dintre d'aquest mètode es crea el *Intent* perquè executi una nova vista.

```
Button btn_config = (Button) findViewById(R.id.btn_config);

btn_config.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
        Intent i = new Intent(getApplicationContext(),Preferences.class);
        startActivity(i);
        }
});
```

Fig. 5.60 Botó de configuració utilitzant la crida Intent.

5.5.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica Inici sessió, Identificació LinkedIn i Menú, presentats anteriorment a la **figura 5.58**.

Vegeu la **Fig. I.1** Vista d'inici, iOS, la **Fig. I.2** Accés a l'aplicació, Android i iOS respectivament, la **Fig. I.3** Autentificació a LinkedIn, Android i iOS respectivament i la **Fig. I.4** Vista del Menú, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.5.3.1. Inici sessió

Aquesta vista conté dos botons creats a l'*Interface Builder* i enllaçats cadascun amb dos *IBOutlets*. Cada botó té associada una *IBAction*. Al prémer el botó de convidat es mostra un *UILabel* (caixa de text) de benvinguda i automàticament es canvia a la vista *Menú*.

5.5.3.2. Identificació LinkedIn

Aquesta vista bé implementada per la llibreria de LinkedIn i és crida des de la vista d'Accés quan es prem el botó de LinkedIn

```
- (IBAction)linkedInLogIn:(id)sender {
    RDLinkedInAuthorizationController* controller = [RDLinkedInAuthorizationController
        authorizationControllerWithEngine:self.engine delegate:self];
…
```

Fig. 5.61 *IBAction* del botó LinkedIn

Un cop autentificat l'usuari es torna a la vista d'*Accés* i es realitza una connexió al servidor enviant el nom i identificador d'usuari i de LinkedIn a la base de dades de l'Escola a través de la funció i URL [102] del servidor *getEetacPerson* per validar si l'usuari és un estudiant o antic alumne de l'EETAC. Aquestes dades s'envien utilitzant la llibreria de JSON [92].

```
- (void)linkedInEngine:(RDLinkedInEngine *)engine requestSucceeded:(RDLinkedInConnectionID
*)identifier withResults:(id)results {
    if( identifier == self.fetchConnection ) {
        linkedinDictionary = (NSDictionary*)results;
        NewID *firstID = [[NewID alloc] initWithEetacID:@"" withLinkedInID:
                [linkedinDictionary objectForKey:@"id"] withName:[linkedinDictionary
                objectForKey:@"first-name"] withSurname:[linkedinDictionary
                objectForKey:@"last-name"]];

        JSON *json = [[JSON alloc] init];
        json.jsonDelegate = self;
        [json postAsyncronousJsonFromNSDictionary:firstID.getNewIDDictionary
                withPath:@"getEetacPerson"];
…
```

Fig. 5.62 Recull de dades de LinkedIn i connexió al servidor

Si l'usuari és validat correctament es rebran totes les dades que el servidor disposa sobre aquest usuari (*EetacPerson*) i es tornarà a fer una connexió amb el servidor a la URL *updateEetacPerson* per enviar la informació completa obtinguda de LinkedIn actualitzar l'usuari. A continuació, es mostrarà un missatge de benvinguda i automàticament s'anirà al *Menú*. Sinó l'usuari no queda validat, es desconnectarà de LinkedIn i romandrà a la vista d'accés amb la possibilitat d'entrar com a convidat.

5.5.3.3. Menú

El menú consta de sis botons, cada un amb els seus enllaços (IBOutlets) i les seves accions (*IBActions*). Al prémer qualsevols dels botons, es carregarà la corresponent vista de l'apartat seleccionat.

El fons del menú té un sistema d'imatges rotatòries obtingudes a través d'una connexió al servidor utilitzant l'adreça *getMenuBackgrounds*. Cada imatge també funciona com a botó i conté un enllaç a una adreça web. D'aquesta manera el fons permet donar notícies o informar d'esdeveniments en forma d'imatge.

5.6. Implementació del mòdul L'Escola

En aquest apartat s'explicarà el funcionament del mòdul L'Escola, i també es veuran les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.6.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes de l'apartat L'Escola i la seva interacció amb el servidor "Sóc de l'EETAC".

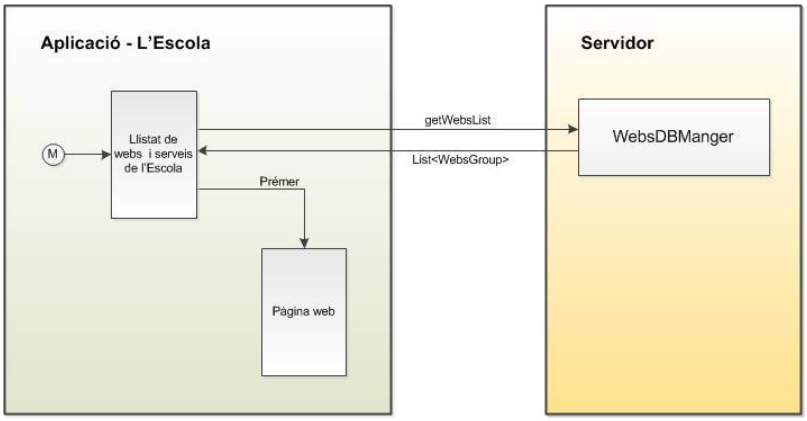

Fig. 5.63 Diagrama de funcionament de l'apartat L'Escola

Quan s'accedeix a l'apartat L'Escola, es carrega automàticament la vista *Llistat de webs i serveis de L'Escola*. Aquest apartat és un llistat amb diferents entrades de pàgines web de l'EETAC i està dividit en tres seccions.

La primera secció és la d'intranets i conté les següents entrades:

- **Alumni:** és el portal per a titulats i titulades de l'EETAC.
- **Atenea:** és la web del Campus Virtual de l'UPC.
- **CBL:** és el portal del Campus del Baix Llobregat de l'UPC.
- **K2pim:** és la web per accedir al servei de correu electrònic de l'UPC.
- **e-Secretaria:** és la web de la Secretaria Acadèmica de l'UPC.
- **SIA:** és el portal del Sistema d'Informació Acadèmica.

La segona secció conté d'informació sobre l'Escola:

- **EETAC:** és la web de l'Escola d'Enginyers de Telecomunicació i Aeroespacial de Castelldefels.
- **Coneix l'EETAC:** és la web amb informació de l'EETAC per a nous estudiants.
- **Bits que volen:** és un blog d'informació i divulgació d'activitats.
- **L'EETAC multimèdia:** web on apareixen vídeos i fotografies de l'Escola.
- **EETAC Apps:** és la web per descarregar altres aplicacions per a mòbils.
- **Recull Notícies:** aquesta web fa un recull de notícies, actes, conferències i altres activitats.
- **RESA:** és la web amb tota la informació sobre la residència d'estudiants Pius Font i Quer.
- **Tour Virtual:** a aquesta web es pot fer un passeig virtual per l'Escola.

La tercera secció està dedicada a tots els màsters que l'Escola imparteix:

- **MASTEAM:** Master of Science in Telecomunication Engineering and Management.
- **Master's degree in Airport and Air Navigation.**
- **GEONA:** Master's degree in Geomatics and Navigation.
- **Master in Smart Healt Innovation.**
- **MAST:** Master in Aerospace Science and Technology.

Quan premem qualsevol de les entrades d'aquestes tres seccions es mostra directament la pàgina web que s'ha sol·licitat. En el cas d'Android es salta a un navegador web fora de l'aplicació, mentre que en el cas d'iOS [86] s'obre una vista nova amb el contingut de la pàgina web.

5.6.2. Android

A continuació es presentaran les implementacions de la interfície gràfica Llistat de webs i serveis de l'Escola i Pàgina web, presentats anteriorment a la **figura 5.63**.

Vegeu la **Fig. I.5** Llistat de pàgines web i serveis, Android i iOS respectivament, i la **Fig. I.6** Exemple d'una pàgina web, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.6.2.1. Llistat de pàgines web i serveis

Durant al llarg de l'aplicació el sistema per crear llistes ha sigut molt semblant. Primer de tot cal tenir la informació amb la qual s'emplenarà la llista, i aquesta informació procedeix del servidor.

En aquest cas el servidor ens proporciona, una descripció curta, la direcció de la pagina web, i la direcció web de la imatge que es mostra.

```
new FetchImageTask() {
        protected void onPostExecute(Bitmap result) {
                    if (result != null) {
                        holder.imgTwitter.setImageBitmap(getRoundedCornerBitmap(result, 5));
                    }
                }
            }.execute(websInput.get(position).getUrlImage());
…
public class FetchImageTask extends AsyncTask<String, Integer, Bitmap> {
        protected Bitmap doInBackground(String... arg0) {
                Bitmap b = null;
                try {
                b = BitmapFactory.decodeStream((InputStream) new URL(arg0[0]).getContent());
                } catch (MalformedURLException e) {
                e.printStackTrace();
                } catch (IOException e) {
                e.printStackTrace();
                }
                return b;
            }
        }
```

Fig. 5.64 Obtenció d'imatges des d'una direcció web

5.6.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica Llistat de webs i serveis de l'Escola i Pàgina web, presentats anteriorment a la **figura 5.63**.

Vegeu la **Fig. I.5** Llistat de pàgines web i serveis, Android i iOS respectivament, i la **Fig. I.6** Exemple d'una pàgina web, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.6.3.1. Llistat de pàgines web i serveis

Aquesta vista consta d'una *UITableView*, aquest element consta de varies funcions al ser interactiu. Al carregar la vista a través de viewDidLoad, es realitza una connexió al servidor automàticament a la URL *getWebList*. Un cop processades les dades utilitzant la llibreria de JSON [92] es mostren a la taula

en forma de cel·les. Al prémer una cel·la s'activa una acció que realitza un salt a la vista *Pàgina web* passant a aquesta nova vista la URL de la pàgina web que es vol consultar.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return [websGroupsList count];
}
```

Fig. 5.65 Retorna el nombre de seccions de la taula

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
        return [[[websGroupsList valueForKeyPath:@"groupLength"] objectAtIndex:section]
                integerValue];
}
```

Fig. 5.66 Retorna el nombre cel·les de cada secció

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    static NSString *CellIdentifier = @"WebListCell";
    WebListTableViewCell *cell = [tableView
        dequeueReusableCellWithIdentifier:CellIdentifier];

    cell.imageView.layer.cornerRadius = 8.0;
    cell.imageView.layer.masksToBounds = YES;
    cell.imageView.layer.borderWidth = 2.0;
    cell.imageView.layer.borderColor = [[UIColor colorWithRed:1.0 green:1.0 blue:1.0
        alpha:0.3] CGColor];

     NSArray *websInput = [[NSArray alloc] initWithArray:[[websGroupsList
valueForKeyPath:@"websGroup"] objectAtIndex:indexPath.section]];

    NSURL *imageURL = [NSURL URLWithString:[[websInput objectAtIndex:indexPath.row]
objectForKey:@"urlImage"]];
    NSData *imageData = [NSData dataWithContentsOfURL:imageURL];
    cell.imageView.image = [UIImage imageWithData:imageData];
    cell.description.text = [[websInput objectAtIndex:indexPath.row]
objectForKey:@"description"];

    return cell;
}
```

Fig. 5.67 Retorna el contingut de cada cel·la

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    WebViewController* wvc = [self.storyboard instantiateViewControllerWithIdentifier:
        @"WebViewController"];

    wvc.title = [[[websGroupsList valueForKeyPath:@"websGroup" ]
        objectAtIndex:indexPath.section] objectForKey:@"navigationTitle"];
    wvc.webURL = [[[websGroupsList valueForKeyPath:@"websGroup" ]
```

```
        objectAtIndex:indexPath.section] objectForKey:@"webURL"];
    }

    [self.navigationController pushViewController:wvc animated:YES ];
    [websListTableView deselectRowAtIndexPath:indexPath animated:YES];
}
```

Fig. 5.68 Envia la URL i carrega la nova vista al prémer qualsevol cel·la

5.6.3.2. Pàgina web

Aquesta vista està formada per un *UIWebView* juntament amb una barra a la part inferior (*UIToolbar*) amb les els botons (*UIBarButtonItem*) per anar endarrere, endavant, parar o tornar a carregar la pàgina web. A l'aparèixer la vista i a través de la funció *viewWillAppear* es carrega la pàgina web a l'*UIWebView*.

```
- (void)viewDidAppear:(BOOL)animated {
    [self.webView loadRequest:[NSURLRequest requestWithURL:[NSURL URLWithString:webURL]]];
}
```

Fig. 5.69 Carrega de la pàgina web seleccionada a la vista anterior

5.7. Implementació del mòdul Missatges

En aquest apartat s'explicarà el funcionament del mòdul Missatges, i també es veuran les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.7.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes de l'apartat Missatges i la seva interacció amb el servidor "Sóc de l'EETAC" i la API [68] de Twitter.

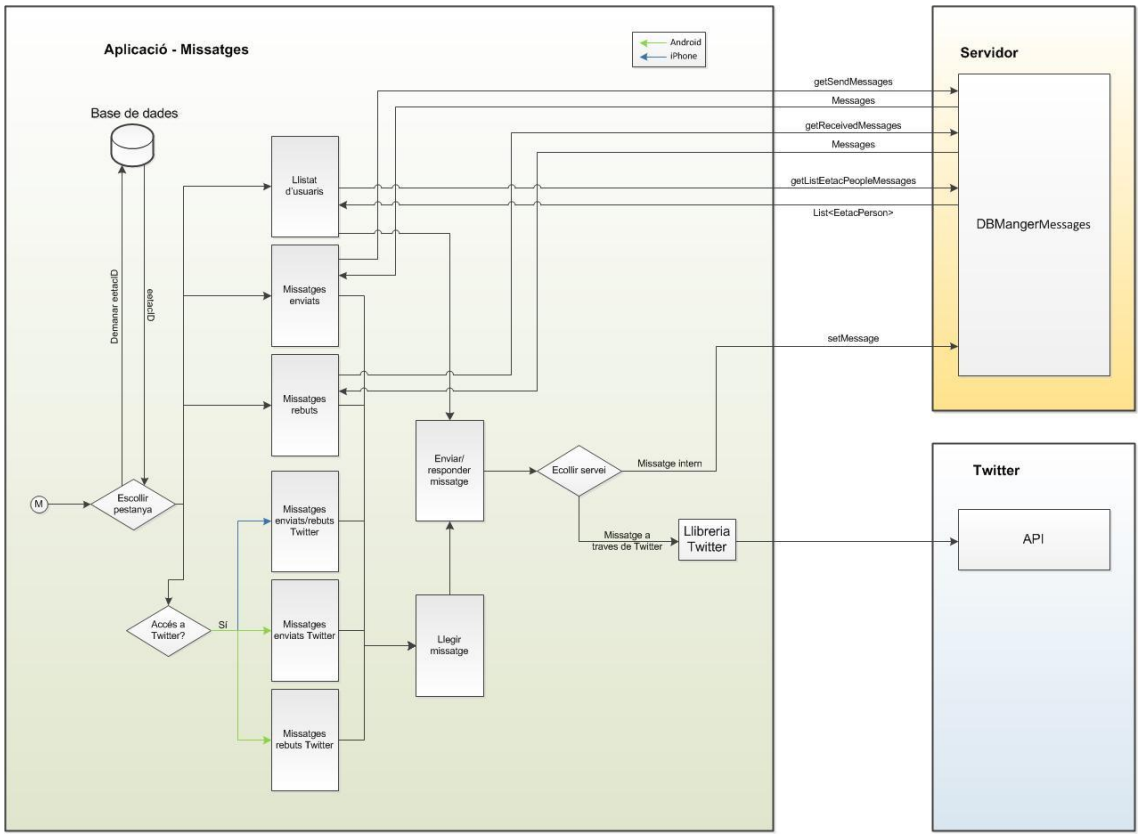

Fig. 5.70 Diagrama de funcionament de l'apartat Missatges

Quan s'accedeix a l'apartat de missatges, podem trobar quatre pestanyes que ens porten a diferents vistes: Escriu (*Llistat d'usuaris*), Enviats (*Missatges enviats*), Rebuts (*Missatges rebuts*) i una sola pestanya per Twitter (*Missatges enviats/rebuts Twitter*) en el cas d'iOS [86] i dues pestanyes (*Missatges enviats Twitter i Missatges rebuts Twitter*) en el cas d'Android [66]. Per defecte en els dos casos s'entra a la pestanya *Missatges rebuts*.

La primera pestanya (*Llistat d'usuaris*) es mostra un llistat de tots els usuaris amb la seva imatge de perfil de LinkedIn, en cas de tenir-la, i el seu nom i cognoms. Aquest llistat només mostra aquells usuaris que poden rebre missatges i que prèviament han autoritzat la recepció de d'aquest. Al prémer a sobre de qualsevol usuari l'aplicació ens desplaçarà a una nova vista on podrem escriure un missatge (*Enviar/respondre Missatge*) tot indicant un assumpte i/o text. Un es tingui el missatge llest es podrà enviar a través del servei de missatges de l'aplicació, i en el cas que l'usuari disposi d'una compte de Twitter i la tingui enllaçada a l'aplicació també podrà enviar el missatge a través de la xarxa social de Twitter, o ambdós serveis a la vedada.

La segona pestanya (*Missatges enviats*) es mostra el llistat de missatges enviats per l'usuari de l'aplicació. A cada missatge d'aquesta llista es pot veure el nom i cognoms, la imatge de perfil de l'usuari a qui s'ha enviat el missatge, juntament amb l'assumpte i l'hora d'enviament. Al prémer qualsevol dels missatges s'entra a una vista molt semblant la que ens permet escriure

missatges (*Llegir missatge*). Des d'aquesta vista també es tindrà l'oportunitat de tornar a enviar un altre missatge al mateix destinatari utilitzant la vista *Enviar/Respondre missatge*.

La tercera pestanya (*Missatges rebuts*) tenim el mateix format de llista de missatges que la segona pestanya, però en aquest cas es mostren els missatges rebuts. Al prémer qualsevols dels missatges s'obrirà la mateixa vista per llegir un missatge (*Llegir missatge*). I des d'aquesta es tindrà l'oportunitat de respondre el missatge utilitzant la vista *Enviar/Respondre missatge*.

Per últim, la quarta pestanya en el cas d'iOS, o la quarta i cinquena pestanya en el cas d'Android, es mostra un llistat de missatges enviats (tuits) i un llistat de missatges rebuts (esments) amb el mateix format que les pestanyes anteriors, però en aquest cas es mostra la imatge i el nom d'usuari del perfil de Twitter. Al prémer cadascun dels tuits o esments es mostrarà la mateixa vista per llegir un missatge (*Llegir missatge*) i també es tindrà l'oportunitat de reenviar un tuit [101] o respondre un esment a través de la xarxa social de Twitter.

5.7.2. Android

A continuació es presentaran les implementacions de la interfície gràfica Llistat d'usuaris, Llistat de missatges enviats i rebuts, i Llistat de tuits enviats i rebuts, presentats anteriorment a la **figura 5.70**.

Vegeu la, **Fig. I.7** Llistat d'usuaris per enviar missatges, Android i iOS respectivament, la **Fig. I.8** Vista per enviar missatges, Android i iOS respectivament, la **Fig. I.9** Llistat de missatges enviats, Android i iOS respectivament, la **Fig. I.10** Vista d'un missatge enviat o rebut, Android i iOS respectivament, **Fig. I.11** Llistat de tuits i mencions, Android, **Fig. I.13** Vista d'un tuit o menció, Android i iOS respectivament, i la **Fig. I.14** Respondre una menció, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.7.2.1. Llistat d'usuaris

Al entrar a la pestanya de llistat d'usuaris el que fa l'aplicació és enviar al servidor un objecta del tipus *EetacPerson* amb l'eetacID, així evitem que el servidor ens enviï a nosaltres mateixos. La resposta del servidor serà enviar un llistat de *EetacPerson*.

La forma de mostra la informació és la mateixa que en el cas del apartat l'Escola, el que cal modificar és el *Adapter* que és l'encarregat de connectar les dades del llistat amb la part visual.

Al prémer sobre qualsevol element del llistat l'aplicació obrirà una nova vista, la de enviar missatges. En ella en cas que l'usuari tingui la imatge personal compartida es veurà la mateixa, seguit del seu nom i cognoms i al igual que la imatge si te compartit el nom d'usuari de Twitter es mostra a continuació del nom.

tenim dos entrades de text, una per l'assumpte i l'altre per el contingut del missatge, com a mínim s'ha d'emplenar una de les dos entrades de text, en cas contrari al prémer al boto d'enviar el missatge ens apareixerà un missatge demanant que s'empleni un dels dos camps.

Com ja s'ha comentat, l'aplicació permet enviar missatges a traves de la missatgeria interna o utilitzant el Twitter, per defecte apareix marcada l'opció d'enviar a traves de la missatgeria interna, però si l'usuari marca la opció del Twitter la entrada de text referent al assumpte s'emplenarà de forma automàtica amb el *hashtag* [82] *#socdelEETAC* seguit del nom d'usuari de Twitter del destinatari, a part d'aquest fet, la entrada quedarà inhabilitada i l'usuari no podrà canviar-la. A més, l'entrada de text del missatge veurà reduïda la quantitat de caràcters que podrà realitza ja que Twitter nomes permet missatges de 140 caràcters, un cop assolit aquest nombre, l'usuari no podrà escriure més. Així doncs, el *hashtag*, més el nom d'usuari del destinatari i el cos del missatge no podran superar els 140 caràcter. Aquests fets queden anul·lats en cas que l'usuari decideixi desmarca l'opció de Twitter.

5.7.2.2. Llista de missatges enviats i rebuts

Tant el llistat de missatges enviats com el de rebuts són pràcticament iguals el que canvia és la font d'informació. Per el cas de missatges enviats l'aplicació realitza una connexió amb la direcció del servidor *getSendMessages* la qual primerament s'ha li passa un objecte *RequestMessages*, que porta informació del usuària que fa la petició, i ens retorna una llista dels missatges enviats.

Els rebuts la connexió amb el servidor és a la direcció *getRecivedMessages* enviant-li al igual que en el cas del enviats un objecte *RequestMessages* .

Una altre diferencia és quan al prémer un element del llistat, s'obre una nova vista on es pot llegir el missatge amb més comoditat, però nomes amb el cas de missatges rebuts apareix el boto de respondre, que la seva funció és obrir la vista per enviar un missatge. Per realitzar aquest procés cal que al fer la crida a la nova vista es passi informació extra, en aquest cas un valor numèric

```
Intent i = new Intent(getApplicationContext(),ChatRes.class);
i.putExtra("case", 2);
startActivity(i);
```

Fig. 5.71 Afegint informació extra la *Intent*.

En la següent vista fem la comprovació de la variable i en cas de que sigui igual a dos, es realitzen alguns canvis, com ara treure el boto de resposta o modificar el text "per a" per el text "de".

```
if (cas==2) {
        btn_reply.setVisibility(View.GONE);
        pera1 = (TextView) findViewById(R.id.tvpera);
        pera1.setText("de");
        pera2.setText(status.getUser().getName());
}
```

Fig. 5.72 Modificació de la vista de lectura de missatges

5.7.2.3. Llista de tuits enviats

Tant el llistat de tuits enviats com el de rebuts nomes serà visible en cas que l'usuari hagi vinculat l'aplicació amb el Twitter.

Per el cas de tuits enviats el primer que realitza l'aplicació per aconseguir el llistat de tuits enviats és identificar al usuari amb el Twitter. Ja que el que fem al enviar un missatge a traves del Twitter és fer un tuit amb un *hashtag* [82] predeterminat, aprofitem aquest fet per després filtra els tuits i mostra els que ens interessen.

Descarreguem el que la llibreria Twitter4j anomena *UserTimeline,* que no és més que un llistat de tots els tuits que ha realitzat l'usuari. Tot seguit crearem un nou llistat que gracies a un bucle anirem omplint únicament amb els tuits que continguin al *hashtag* desitjat, en aquest cas *#socdelEETAC.*

Ara ja es pot visualitzar el llistat de tuits enviats. Cada element de la llista esta compost per la imatge de la comte d'usuari del Twitter, el nom, el nom de la compte d'usuari, el missatge (dos línies com a màxim), la data i hora en que es va fer el tuit. Al prémer qualsevol element s'obre una altre vista on es far més còmode la visualització del missatge.

5.7.2.4. Llista de tuits rebuts

Aquest cas és molt similar al de tuits enviats. També cal estar identificat, i aquí no descarreguem el *UserTimeline* sinó que descarreguem un llistat anomenat *Mentions* (mencions). Les mencions són la forma que te el Twitter d'informar al usuari que algú ha escrit un tuit on en el contingut del missatge apareix el seu nom d'usuari precedit d'una arrova.

Ara com en cas anterior filtrem el llistat de missatges a traves del *hashtag* per mostrar únicament els missatges del nostre interès.

Un altre element diferenciador, és que quan premem un algun dels elements per ampliar la informació, l'aplicació ens enviarà a la vista d'enviar missatges, anteriorment descrita, on podrem realitzar un tuit amb el comentat *hashtag* i fen menció al usuari que ens ha escrit. En aquesta ocasió la opció d'enviar un missatge utilitzant la missatgeria interna no serà possible ja que no es disposarà del *eetacID.*

5.7.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica Llistat d'usuaris, missatges enviats i rebuts, tuits enviats i rebuts, llegir missatge i enviar / respondre missatge presentats anteriorment a la **figura 5.70**.

Vegeu la **Fig. I.7** Llistat d'usuaris per enviar missatges, Android i iOS respectivament, la **Fig. I.8** Vista per enviar missatges, Android i iOS respectivament, la **Fig. I.9** Llistat de missatges enviats, Android i iOS respectivament, la **Fig. I.10** Vista d'un missatge enviat o rebut, Android i iOS respectivament, **Fig. I.11** Llistat de tuits i mencions, Android, la **Fig. I.12** Llistat de tuits i mencions, iOS, **Fig. I.13** Vista d'un tuit o menció, Android i iOS respectivament, i la **Fig. I.14** Respondre una menció, Android i iOS respectivament per observar l'aspecte de la interfície gràfica un cop implementada.

5.7.3.1. Llistat d'usuaris, missatges enviats i rebuts, i tuits enviats i rebuts

Aquestes quatre vistes en realitat només és està formada per una. La vista està formada per un taula (*UITableView)* i una barra (*UIToolbar)* a la part inferior. Dins de la barra inferior hi ha l'element que permet visualitzar diferents vistes dins de la mateixa, són els segments o pestanyes (*UISegmentedControl*). Aquest element s'encarrega de saber quin segment es prem (*IBAction*) i a través d'aquesta variable es poden mostrar valors diferents dins d'una mateixa taula. Un altre element important són les cel·les (*UITableViewCell*), dins d'una mateixa taula es poden configurar diferents cel·les, cada una amb diferents elements visuals i configurades a través d'una classe pròpia. Amb aquest dos elements es crea l'efecte d'estar canviant de vista. Per obtenir les dades necessàries per les vistes *Llista d'usuaris*, *missatges enviats* i *missatges rebuts* realitzarem una connexió al servidor a través de les URL *getListEetacPeopleMessages*, *getSendMessages* i *getReceibedMessages* respectivament.

```
- (IBAction)changeOptions:(id)sender {
    switch (messagesSegmentControl.selectedSegmentIndex) {
        case 0: {
            JSON *json = [[JSON alloc] init];
            [json setJsonDelegate:self];
            [json postAsyncronousJsonFromNSDictionary:
```

```
                [me getEetacIDDictionaryFromEetacPerson:me]
                withPath:@"getListEetacPeopleMessages"];
            break;
        }
        case 1: {
            RequestMessages *requestSendMessages = [[RequestMessages alloc]
                initWithEetacID:me.eetacID withPostPosition:0 withMessageID:0];
            JSON *json = [[JSON alloc] init];
            [json setJsonDelegate:self];
            [json postAsyncronousJsonFromNSDictionary:[requestSendMessages
                getRequestMessagesDictionaryFromRequestMessages:requestSendMessages]
                withPath: @"getSendMessages"];
            break;
        }
…
```

Fig. 5.74 Canvi de segment (vista) i petició de dades a través de JSON

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath*)indexPath {
    static NSString *CellIdentifier = @"EetacPeopleMessagesCell";
    EetacPeopleMessagesTableViewCell *peopleCell =
        [tableView dequeueReusableCellWithIdentifier:CellIdentifier];

    static NSString *CellIdentifierMessages = @"MessagesCell";
    MessagesTableViewCell *messageCell =
        [tableView dequeueReusableCellWithIdentifier:CellIdentifierMessages];

    static NSString *CellIdentifierTweets = @"TweetCell";
    SocialTableViewCell *tweetCell =
        [tableView dequeueReusableCellWithIdentifier:CellIdentifierTweets];
    …
    switch (messagesSegmentControl.selectedSegmentIndex) {
        case 0: { … }
        case 1: {
            Message *aMessage = [listItems objectAtIndex:indexPath.row];
            if (aMessage.uiImage != nil) {
                messageCell.imageView.image = aMessage.uiImage;
            }
            messageCell.titleLabel.text = aMessage.destinationStudent;
            messageCell.subjectLabel.text = aMessage.subject;
            messageCell.dateSubject.text = aMessage.date;
            return messageCell;
            break;
        }
        …
```

Fig. 5.74 Definició de les diferents cel·les i canvi de cel·la visualitzada a la taula segons el segment seleccionat

5.7.3.2. Llegir missatge

Aquestes vista està composta de varies caixes de text (*UILabels*) per mostrar el contingut d'un missatge o tuit [101] (nom i cognoms o nom d'usuari, data i hora, assumpte); el cos del missatge es un altre tipus de caixa de text (*UITextView*), aquest element permet detectar enllaços, números de telèfon...; també conté una imatge (*UIImage*) per mostrar la imatge de perfil de l'usuari i un botó

(*UIButton*) per respondre el missatge visualitzat. Al prémer el botó per respondre es carregarà la vista *Enviar/respondre missatge*.

5.7.3.3. Enviar / respondre missatge

La única diferència entre aquesta vista i l'anterior només radica en els elements que mostren l'assumpte i el cosa del missatge, aquest elements en aquesta vista són caixes de text editables, *UITextField* en el cas de l'assumpte i *UITextView* per al cos del missatge. Al prémer el botó enviar (*IBAction*) apareixerà un menú desplegable amb tres opcions: enviar el missatge a l'EETAC, a Twitter o a l'EETAC+Twitter, si l'altre usuari no té compte de Twitter les dues últimes opcions no estaran disponibles. Segons l'opció que es premi s'executarà una funció o altra. Per enviar missatges a través de l'EETAC, es realitzarà una connexió al servidor a través de la URL [102] *setMessage*. Per obtenir i enviar tuits utilitzarem la llibreria i la API [68] de Twitter.

```
UIActionSheet *sendMessageSheet = [[UIActionSheet alloc]
        initWithTitle:@"Envia el missatge a" delegate:self
        cancelButtonTitle:@"Cancel·la"
        destructiveButtonTitle:nil
        otherButtonTitles:@"Jo sóc de l'EETAC", @"Twitter", @"Twitter + EETAC", nil];
[sendMessageSheet showInView:self.view];
```

Fig. 5.75 Visualitzar el menú per enviar un missatge

```
- (void)sendMessageToImFromEetac {
    Message *aMessage = [[Message alloc] initWithEetacID:myEetacID
        withDestinationEetacID:destinationEetacID withSubject:subjectTextField.text
        withDate:dateLabel.text withMessage:messageTextView.text];
    JSON *json = [[JSON alloc] init];
    [json setJsonDelegate:self];
    [json postAsyncronousJsonFromNSDictionary:[aMessage
        getMessagesDictionaryFromMessages:aMessage] withPath:@"setMessage"];
}
```

Fig. 5.76 Enviar missatge a través de l'EETAC

```
- (void)sendMessageToTwitter {
    NSString *hashTags = @"#socdelEETAC";
    NSString *tweetSubject = [NSString stringWithFormat:@"%@. ", subjectTextField.text];
    NSString *tweetMessage = messageTextView.text;
    NSString *tweet = [NSString stringWithFormat:@"%@ @%@ %@%@", hashTags,
        destinationTwitterName, tweetSubject, tweetMessage];
    NSString *tweetURLString = [NSString stringWithFormat:
        @"http://api.twitter.com/1/statuses/update.json"];
    NSURL *tweetURL = [NSURL URLWithString:tweetURLString];
    TWRequest *request = [[TWRequest alloc] initWithURL:tweetURL parameters:[NSDictionary
        dictionaryWithObject:tweet forKey:@"status"] requestMethod:TWRequestMethodPOST];
```

```
    …
    request.account = twitterAccount;
    [request performRequestWithHandler:^(NSData *responseData, NSHTTPURLResponse
        *urlResponse, NSError *error) {
…
```

Fig. 5.77 Enviar missatge a través de Twitter

5.8. Implementació del mòdul Companys

En aquest apartat s'explicarà el funcionament del mòdul Companys, així com les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.8.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes de l'apartat Companys i la seva interacció amb el servidor "Sóc de l'EETAC".

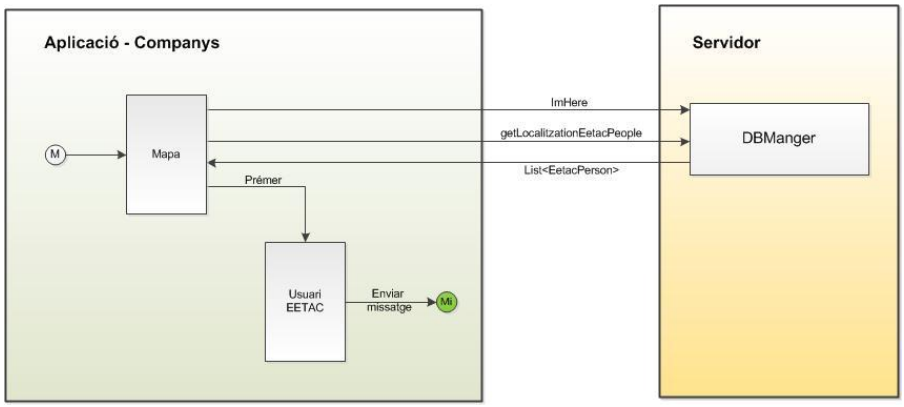

Fig. 5.78 Diagrama de funcionament de l'apartat Companys

Quan s'accedeix a l'apartat Companys es mostra un planell (*Mapa*), per defecte aquest planell es centra a les coordenades de l'EETAC. En aquest mapa es mostraran tots els usuaris de l'aplicació que permetin la seva localització i que estiguin dins de l'àrea mostrada pel mapa, també es mostrarà la localització de les empreses d'aquests usuaris. Al mateix mapa també es podrà mostrar la nostra localització en cas de prémer el botó de localització.

Al prémer sobre un usuari apareixerà una petita vista on es mostrarà la imatge i el nom i cognoms del perfil de LinkedIn. A través d'aquesta petita vista es podrà accedir a una altre (*Usuari EETAC*) amb informació més completa sobre l'usuari seleccionat al mapa. La informació que es mostra a *Usuari EETAC*

vindrà determinada per les autoritzacions que l'usuari que s'ha seleccionat hagi permès mostrar i està aquesta informació està dividida en quatre seccions: dades personals, dades universitàries, feina actual i feines passades.

En el cas que aquest usuari no ens restringeixi l'accés a cap informació a la secció de dades personals es mostrarà la imatge, nom i cognoms del perfil de LinkedIn i un botó per enviar-li un missatge. La secció dades universitàries mostrarà el correu electrònic d'estudiant de l'UPC, la titulació i la data de finalització de la titulació. La tercera i quarta secció, tal com els seus noms indiquen, es mostrarà un llistat de les feines actuals i passades del perfil de LinkedIn de l'usuari seleccionat. A cada feina es mostrarà el nom de l'empresa, el càrrec que l'usuari ha tingut dins d'aquesta empresa i el període durant el qual ha estat realitzant aquesta feina. En que es disposi de les dades de localització d'una empresa es podrà prémer el nom de l'empresa per localitzar-la a través de la vista anterior *Mapa*. En el cas que es vulgui enviar un missatge a l'usuari seleccionat, al prémer el botó es saltarà directament a la vista *Enviar/respondre Missatge* de l'apartat Missatges.

5.8.2. Implementació Android

A continuació es presentaran les implementacions de la interfície gràfica Mapa i Usuari EETAC presentats anteriorment a la **figura 5.78**.

Vegeu la **Fig. I.15** Mapa amb companys, Android i iOS respectivament, i la **Fig. I.16** Informació d'un company, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.8.2.1. Mapa

A l'apartat mapes és un dels llocs on es treu més partit a la llibreria *GreenDroid*, ja que ocupo els tres botons que m'ofereix la llibreria juntament amb submenús.

El primer boto, començant per l'esquerra, desplega un submenú amb dos opcions més, Titulats a prop i La meva ubicació.

Al prémer la primera opció el primer que fa l'aplicació és comprovar si el usuari te algun sistema de localització connectat, en cas negatiu apareixerà un missatge informant al usuari, però en cas positiu, l'aplicació utilitzarà el que doni una precisió més elevada. Hi ha disponible dos sistemes, per antenes de telefonia o per *GPS,* i tot i que el GPS és molt més precís també és més lent a la hora de localitzar la posició.

Un cop aconseguida la posició (longitud i latitud) el pas següent és aconseguir la distancia en metres, de la part central del mapa (posició del usuari) i la cantonada esquerra de la pantalla, aquesta longitud variarà en funció del nivell de zoom que l'usuari tingui en el mapa. Aquesta distancia serà útil perquè el

servidor nomes ens retorni els titulats puguem visualitzar a la pantalla del dispositiu, rebent sempre els d'usuaris "útils".

El que fa l'aplicació a continuació és emplenar el objecta *ImHere*, que serà el que s'envia al servidor.

```
ImHere imHere = new ImHere();
imHere.setMyLat(lat);
imHere.setMyLon(lon);
imHere.setMapDistance(distance);
credentials = new Credentials(sharedPrefs);
imHere.setEetacID(credentials.getEetacID());
```

Fig. 5.79 Emplenant objecte *ImHere*.

La resposta del servidor al enviar un *ImHere* és la de un llistat de *EetacPerson,* es realitza un bucle i es posiciona cada persona en funció de la seva latitud i longitud.

```
mapOverlays = mapView.getOverlays();
itemizedOverlay = new CustomItemizedOverlay<CustomOverlayItem>(drawable, mapView);
for (EetacPerson temp : persones) {
        lat = temp.getLat();
        lon = temp.getLon();
        if (lat != 0.0 && lon != 0.0) {
                GeoPoint geoPoint = new GeoPoint((int) (lat * 1E6),     (int) (lon * 1E6));
                CustomOverlayItem overlayItem = new CustomOverlayItem(geoPoint,
                temp.getName(), temp.getSurname(), temp.getUrlImageLinkedIn());
                itemizedOverlay.addOverlay(overlayItem);
                mapOverlays.removeAll(mapOverlays);
                mapOverlays.add(itemizedOverlay);
        }
}
```

Fig. 5.80 Posicionant usuaris a la vista de Mapes.

El següent botó de La meva ubicació utilitza un funció de Google que localitza la posició del usuari i posa un punt blau amb un radi de cobertura en funció de la precisió que obtingui a la hora de posicionar.

```
mapView.getOverlays().add(mlocationOverlay);
mlocationOverlay.runOnFirstFix(new Runnable() {
        public void run() {
                mapView.getController().animateTo(mlocationOverlay.getMyLocation());
        }
});
```

Fig. 5.81 Posicionant el usuari a la vista de Mapes.

El següent boto de la vista (el del mig) desplega novament un submenú on es troben tres botons més.

El primer boto és el de satèl·lit i la seva funció és la de posar el mapa en forma d'imatges de satèl·lit. El següent ens permet veure informació referent al transit, marcant les carreteres en diferents color depenent de la fluïdesa de circulació. I l'últim és per posar al mapa en forma d'imatges.

```
public void onQuickActionClicked(QuickActionWidget widget, int position) {
        switch (position) {
                case 0:
                        mapView.setSatellite(true);
                        break;
                case 1:
                        mapView.setTraffic(true);
                        break;
                case 2:
                        mapView.setSatellite(false);
                        break;
                        }
                }
        });
```

Fig. 5.82 Posicionant el usuari a la vista de Mapes.

L'últim botó de la barra, té la mateixa funció que el boto mencionat anteriorment Titulats a prop, amb la diferencia que no s'envia la posició del usuari sinó que s'envia la posició del centre del mapa en el moment de prémer el boto.

Per últim la forma que te Android [66] per ensenya petits missatges informatius és el anomenat Toast. La seva creació és relativament senzilla i en el transcurs de l'aplicació la seva utilització és abundant.

```
Toast.makeText(this, "No hi ha titulats per la zona", Toast.LENGTH_LONG).show();
```

Fig. 5.83 Creació d'un *Toast*

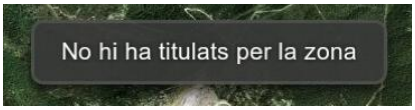

Fig. 5.84 Detall d'un *Toast*

5.8.2.2. Informació d'usuari

Per arribà aquesta vista cal prémer a un usuari en al mapa, on ens apareixerà un globus amb la foto i el nom complet de la persona, al prémer aquest globus saltem a la vists de Informació d'usuari.

Aquesta vista a part de proporcionar la informació que l'usuari vulgui compartir també mosta un boto que carregarà la vista d'enviar missatges.

5.8.3. Implementació iOS

A continuació es presentaran les implementacions de la interfície gràfica Mapa i Usuari EETAC presentats anteriorment a la **figura 5.78**.

Vegeu la **Fig. I.15** Mapa amb companys, Android i iOS respectivament, i la **Fig. I.16** Informació d'un company, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.8.3.1. Mapa

Aquesta vista està formada per un *MKMapView* (*IBOutlet*) encarregat de mostrar el mapa. També està formada per una barra a la part inferior (*UIToolBar*) amb els botons (*UIToolBarItem*) de localització i tipus de mapes. Aquesta vista utilitza la llibreria *Map Kit* ja descrita anteriorment. Per obtenir quins usuaris tenim més a prop s'agafen les coordenades del centre del mapa i el radi en quilòmetres que té l'àrea visualitzada al *MKMapView*. Aquestes dades es guarden en un objecte *ImHere* i s'envien al servidor través d'una connexió a la URL *getLocalitzationEetacPeople*, el servidor ens enviarà un llistat d'*EetacPerson*. Cada cop que es desplaci, s'ampliï o redueixi el mapa es realitzarà la mateixa operació.

```
[mapView setMapType:MKMapTypeStandard];
[mapView setZoomEnabled:YES];
[mapView setScrollEnabled:YES];
MKCoordinateRegion region = { {0.0, 0.0 }, { 0.0, 0.0 } };
region.center.latitude = 41.275365;
region.center.longitude = 1.986921;
region.span.longitudeDelta = 0.005f;
region.span.latitudeDelta = 0.005f;
[mapView setRegion:region animated:YES];
[mapView setDelegate:self];
```

Fig. 5.85 Configuració i càrrega del mapa

```
- (void)getNearestEetacPeople {
    JSON *json = [[JSON alloc] init];
    [json setJsonDelegate:self];
    [json postAsyncronousJsonFromNSDictionary:imHere.getImHereDictionary
        withPath:@"getLocalitzationEetacPeople"];
}
```

Fig. 5.86 Enviament de l'objecte *ImHere*

```
- (void)mapView:(MKMapView *)mV regionDidChangeAnimated:(BOOL)animated {
    [self calculateDistance];
    if (imHere != NULL) {
        imHere.eetacID = me.eetacID;
        imHere.linkedInID = me.linkedinID;
        imHere.lat = mapView.region.center.latitude;
        imHere.lon = mapView.region.center.longitude;
        imHere.distance = myDistance;
        imHere.allowPosition = NO;
        [self getNearestEetacPeople];
    }
}
```

Fig. 5.87 Reenviament de l'objecte *ImHere* al desplaçar el mapa

5.8.3.2. Informació d'usuari

Aquesta vista està formada per una taula (*UITableView*), però a diferència de la taula de la vista *Llista de webs i serveis* aquesta taula està agrupada, el funcionament i la implementació és el mateix, però les cel·les es mostren diferent, agrupades en comptes d'una llista plana. La taula està dividida en quatre seccions, les cel·les de la primera secció mostraran les dades personals on i un botó (UIButton juntament amb una *IBAction*) serà l'encarregat de permetre el salt a la vista *Escriure / respondre missatge*. La segona secció mostrarà les dades universitàries i la tercera i quarta secció mostraran les dades laborals de LinkedIn. Si una empresa que apareix a la tercera o quarta secció tenim les seves coordenades, la cel·la mostrarà una fletxa (accessori) i aquest actuarà com un botó. Al prémer aquest accessori es retornarà al mapa i es mostrà la ubicació de l'empresa.

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    Job *selectedJob = [eetacPerson.currentJobs objectAtIndex:indexPath.row];
    ImHere *companyLocation = [[ImHere alloc] initWithEetacID:@"0" withLinkedInID:@"0"
        withLat:selectedJob.company.companyLat withLon:selectedJob.company.companyLon
        withDistance:0.0 withAllowPosition:NO];
        [delegate returnImHere:companyLocation];
        …
```

Fig. 5.88 Acció per mostrar una empresa al mapa.

5.9. Implementació del mòdul Social

En aquest apartat s'explicarà el funcionament del mòdul Social, així com les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.9.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes corresponents a la visualització de les xarxes socials de l'Escola i la seva interacció amb les llibreries de LinkedIn i Twitter.

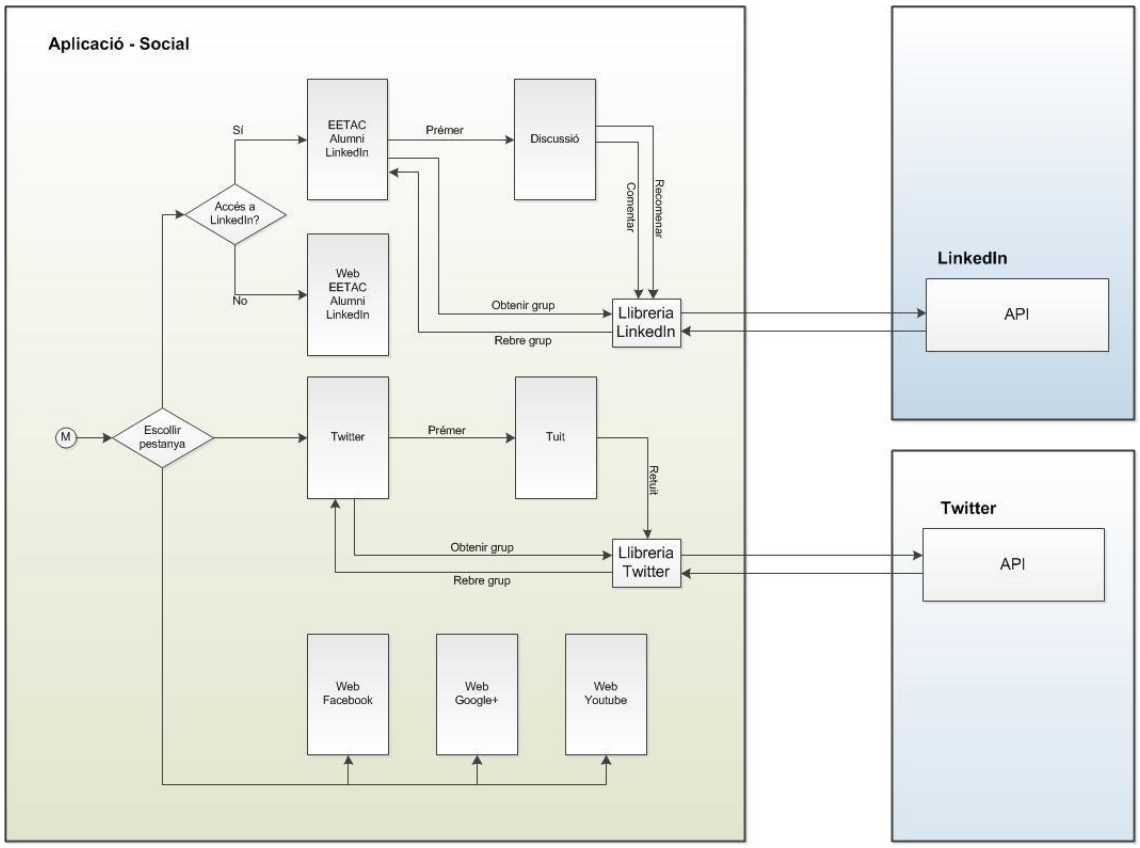

Fig. 5.89 Diagrama de funcionament de l'apartat Social

Quan s'accedeix a l'apartat de missatges, podem trobar cinc pestanyes que ens porten a diferents vistes: LinkedIn (*EETAC Alumni LinkedIn*), *Twitter*, Facebook (*Web Facebook*) Google+ (*Web Google+*) i YouTube (Web YouTube). Per defecte s'entra a la pestanya de LinkedIn.

La primera pestanya permet mostrar el grup EETAC Alumni que l'Escola disposa a la xarxa professional LinkedIn. Primer de tot, es comprova si l'usuari

de l'aplicació està validat a LinkedIn, ja que la llibreria de LinkedIn no permet utilitzar la API [68] si l'usuari no s'ha autentificat. En el cas que l'usuari accedeixi a l'aplicació com a convidat es mostrarà directament la pàgina web del grup EETAC Alumni. En cas contrari, es mostra el llistat de discussions del grup i al prémer una discussió es mostra la informació completa d'aquesta a la vista *Discussió*. A través d'aquesta última vista es podrà recomanar, inclús comentar la discussió utilitzant la llibreria de LinkedIn.

La segona pestanya mostra el llistat de tuits de la compte de Twitter de que disposa l'Escola (@EETAC_UPC). El format de la llista és el mateix de la vista *Missatges enviats/rebuts Twitter* de l'apartat Missatges. Al prémer qualsevol dels tuits de la llista es presentarà una nova vista (*Tuit*) amb el mateix format que la vista *Llegir missatge* de l'apartat Missatges. Des d'aquesta vista es tindrà l'oportunitat de repiular [99] a través de la nostre compte de Twitter el tuit [101] seleccionat.

La tercera, quarta i cinquena pestanya mostren respectivament les pàgina web de Facebook, Google+ i YouTube que l'Escola disposa en aquestes xarxes socials.

5.9.2. Android

A continuació es presentaran les implementacions de la interfície gràfica EETAC Alumni LinkedIn, Twitter presentats anteriorment a la **figura 5.89**.

Vegeu la **Fig. I.17** Llistat de discussions de l'EETAC Alumni a LinkedIn, Android i iOS respectivament, la **Fig. I.18** Vista d'una discussió, Android i iOS respectivament, la **Fig. I.19** Llistat de tuits de L'Escola, Android i iOS respectivament, la **Fig. I.20** Vista d'un tuit, Android i iOS respectivament, la **Fig. I.21** Xarxa social Facebook, Android i iOS respectivament, la **Fig. I.22** Xarxa social Google+, Android i iOS respectivament i la **Fig. I.23** Xarxa de vídeos YouTube, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.9.2.1. EETAC Alumni LinkedIn

El cas de LinkedIn te dues vessants diferents. La primera és en cas que el usuari entri a l'aplicació com a convidat, i que entri havent iniciat sessió a LinkedIn.

Si ha iniciat sessió amb el LinkedIn el que podrà veure el usuari és un llistat de discussions que el grup de l'Escola va realitzant. Tot gracies al ajut de la llibreria ja comentada de LinkedIn-j. Prémer algun element de la llista, l'aplicació obre el que Android anomena *AlertDialog,* que no és més una finestra al estil "pop-up" que permet la lectura completa del missatge i dona l'opció de recomanar la discussió desitjada.

En cas que s'estigui utilitzant l'aplicació en Convidat, al no estar identificat amb el LinkedIn, no es pot descarrega el llistat de discussions, per substituir aquesta cadència es mostra directament la web on es pot trobar el llistat. Amb aquest mètode l'opció, el mètode de recomanació no es pot executar.

5.9.2.2. Twitter

A l'igual que el LinkedIn, gràcies a l'ajuda de la llibreria de Twitter4j, en aquesta pestanya ens descarreguem el llistat de tuits i repiulades que l'Escola realitza. En aquest cas és indiferent si l'usuari té l'aplicació vinclada amb el seu compte de Twitter, el llistat es veu igual. Pel contrari el que podrà realitzar l'usuari, seran repiulades, que es realitzen de la mateixa manera que les recomanacions en el LinkedIn, mitjançant un *AlertDialog.*

5.9.2.3. Altres xarxes socials

La resta de xarxes socials són accessos a les webs que te l'Escola en cada una d'elles. Per realitza aquesta part es crea una única vista igual per les tres xarxes, i quant es crida la vista, s'envia un valor numèric per carga una web especifica (Facebook, Google+, YouTube).

```
switch(webs){
                case 0:
                        myWebView.loadUrl("http://www.linkedin.com/groups/EPSC-Alumni-former-
                        EUPBL-86759?mostPopular=&gid=86759");
                        myWebView.setWebViewClient(new verMiWeb());
                        break;
                case 1:
                        myWebView.loadUrl("https://plus.google.com/107196343161819110849");
                        break;
                case 2:
                        myWebView.getSettings().setJavaScriptEnabled(true);
                myWebView.getSettings().setJavaScriptCanOpenWindowsAutomatically(false);
                        myWebView.loadUrl("http://www.facebook.com/EETAC");
                        break;
                case 3:
                        myWebView.getSettings().setJavaScriptEnabled(true);
                myWebView.getSettings().setJavaScriptCanOpenWindowsAutomatically(false);
                myWebView.loadUrl("http://www.youtube.com/user/upceetac");
                        break;
                }
```

Fig. 5.90 *Switch* / *Case* de selecció de webs

5.9.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica EETAC Alumni LinkedIn i Twitter, Discussió i Tuit, i Facebook, Google+ i YouTube presentats anteriorment a la **figura 5.89**.

Vegeu la **Fig. I.17** Llistat de discussions de l'EETAC Alumni a LinkedIn, Android i iOS respectivament, la **Fig. I.18** Vista d'una discussió, Android i iOS respectivament, la **Fig. I.19** Llistat de tuits de L'Escola, Android i iOS respectivament, la **Fig. I.20** Vista d'un tuit, Android i iOS respectivament, la **Fig. I.21** Xarxa social Facebook, Android i iOS respectivament, la **Fig. I.22** Xarxa social Google+, Android i iOS respectivament i la **Fig. I.23** Xarxa de vídeos YouTube, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.9.3.1. EETAC Alumni LinkedIn i Twitter

Tal com s'ha explicat a l'apartat Missatges, aquestes dues vistes en realitat només és una formada per una taula (*UITableView*) i el contingut de les cel·les (*UITableViewCell*) que es mostra depèn del segment que es tingui seleccionat (*UISegmentControl*) a la barra inferior (*UIToolbar*). Al prémer el segment LinkedIn o Twitter es realitza una connexió a les respectives APIs per obtenir les discussions del grup EETAC Alumni de LinkedIn o els Tuits de l'usuari de Twitter de l'Escola.

5.9.3.2. Discussió i Tuit

Aquestes dues vistes són molt semblants a la vista *Llegir Missatge* de l'apartat Missatges. Són un conjunt de caixes de text (*UILabels*) que permeten mostrar tots els camps necessaris, en el cas d'un tuit són els mateixos mostrats que la vista *Llegir missatge*, la única diferència és el botó (*UIButton*) mostrat, aquest permetrà repiular un tuit en compte de respondre un missatge. En el cas de les discussions la vista és una mica més complexa, ja que mostra els comentaris d'una discussió i la llargada de la vista és variable. Per implementar aquesta part s'ha hagut d'afegir un element que permeti desplaçar la vista amunt i avall en el cas que sigui mes llarga, es tracta d'un *UIScrollView*, també s'han hagut de generar les caixes de text dinàmicament a través de codi, ja que no es pot saber la quantitat de comentaris que pot tenir una discussió.

```
…
for (int i=0; i<[post.comments count]; i++) {
    Comment *comment = [post.comments objectAtIndex:i];
    UIImageView *commentPicture = [[UIImageView alloc] initWithFrame:CGRectMake(20.0, 35.0 +
        commentBlock, 35.0, 35.0)];
    commentPicture.image = comment.picture;
    [commentsView addSubview:commentPicture];
    UILabel *commentName = [[UILabel alloc] initWithFrame:CGRectMake(60.0, 35.0 +
```

```
        commentBlock, 200.0, 35.0)];
    commentName.text = [NSString stringWithFormat:@"%@ %@", comment.firstName,
        comment.lastLame];
    [commentsView addSubview:commentName];
     UITextView *commentText = [[UITextView alloc] init];
     commentText.text = comment.comment;
     commentText.frame = CGRectMake(50.0, commentText.frame.origin.y, 220.0,
        commentText.contentSize.height);
     [commentsView addSubview:commentText];
     …
}
commentsView.frame = CGRectMake(20.0, 20.0 + linkedinView.frame.origin.y +
linkedinView.frame.size.height, 280.0, commentBlock);

viewButton.frame = CGRectMake(20.0, commentsView.frame.origin.y +
commentsView.frame.size.height + 20.0, 280.0, 44.0);
…
linkedinScrollView.contentSize = CGSizeMake(self.view.frame.size.width,
recommendButton.frame.origin.y + recommendButton.frame.size.height + 20.0);
```

Fig. 5.91 Caixes de text dinàmiques i càlcul de la llargada de la vista

5.9.3.3. Facebook, Google+ i YouTube

En aquesta vista, igual que l'apartat L'Escola, està formada per un *UIWebView* juntament amb una barra a la part inferior (*UIToolbar*) amb les els botons (*UIBarButtonItem*) per anar endarrere, endavant, parar o tornar a carregar la pàgina web. A l'aparèixer la vista i a través de la funció *viewWillAppear* es carrega la pàgina web a l'*UIWebView*

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    Job *selectedJob = [eetacPerson.currentJobs objectAtIndex:indexPath.row];
    ImHere *companyLocation = [[ImHere alloc] initWithEetacID:@"0" withLinkedInID:@"0"
        withLat:selectedJob.company.companyLat withLon:selectedJob.company.companyLon
        withDistance:0.0 withAllowPosition:NO];
        [delegate returnImHere:companyLocation];
        …
```

Fig. 5.92 Acció per mostrar una empresa al mapa.

5.10. Implementació del mòdul Dades

En aquest apartat s'explicarà el funcionament del mòdul Dades, i també es veuran les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.10.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes de l'apartat Dades i la seva interacció amb el servidor "Sóc de l'EETAC".

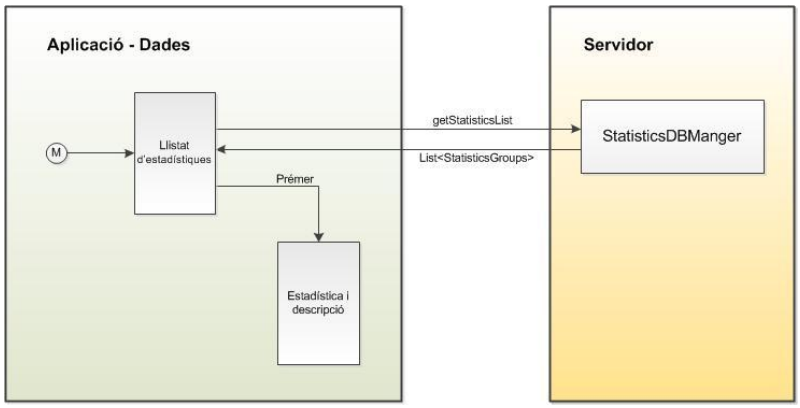

Fig. 5.93 Diagrama de funcionament de l'apartat Dades

Quan s'accedeix a l'apartat Dades, es carrega automàticament la vista *Llistat d'estadístiques*. Aquest apartat és un llistat amb diferents entrades de dades i estadístiques amb informació referent als estudiants i estudis de l'EETAC. Igual que l'apartat L'Escola aquesta llista tindrà varies seccions i cada secció tindrà varies entrades. A cada entrada es visualitzarà un títol i una descripció i al prémer qualsevol de les entrades es presentarà la vista *Estadística i descripció*. Aquesta vista mostrarà un gràfic amb les dades que es volen donar a conèixer i un text explicatiu d'aquestes.

5.10.2. Android

A continuació es presentaran les implementacions de la interfície gràfica Llistat d'estadístiques, i Estadística i descripció, presentats anteriorment a la **figura 5.93**.

Vegeu la **Fig. I.24** Llistat de dades i estadístiques, Android i iOS respectivament, i la **Fig. I.25** Gràfica d'una estadística, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.10.2.1. Llistat d'estadístiques

Aquesta és molt semblant a les comentades fins ara, un element *ListView* que es connecta amb el servidor, rep un llistat i el prémer algun dels camps obre

una nova vista, però te un petit punt diferenciador, i és que conte elements separadors per distingí entre els diferents grups.

Per realització dels elements separadors ha calgut descompondre el llistat rebut del servidor (ja que és un llistat que conte llistats) i reconstruir-lo creant un únic llistat afegint-hi els elements que seran els separadors. Ara cal que a la hora de mostra els elements es distingeix-hi entre separador i estadística i un cop distingit s'esculli l'aparença a mostra.

5.10.2.2. Estadística i descripció

Al prémer qualsevol dels elements del llistat (menys els separadors) s'obrirà una nova vista, on aproximadament la mitat superior es mostra una web (allotjada al servidor de l'escola) amb la gràfica desitjada, i la mitat inferior és una breu descripció de la mateixa.

5.10.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica Llistat d'estadístiques, i Estadística i descripció, presentats anteriorment a la **figura 5.93**.

Vegeu la **Fig. I.24** Llistat de dades i estadístiques, Android i iOS respectivament, i la **Fig. I.25** Gràfica d'una estadística, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.10.3.1. Llistat d'estadístiques

Aquesta vista està implementada igual que la vista *Llistat de web i serveis* de l'apartat Escola, està formada per una taula (*UITableView*), dividida en varies seccions. Al carregar la vista a través de la funció *viewDidLoad* es connectarà al servidor a través de la URL [102] *getStatisticsList* per rebre un llistat d'objectes *StatiscsGroups*, que es correspon amb el número de seccions. Al prémer qualsevols de les cel·les s'accedirà a la vista *Informació de l'estadística*.

5.10.3.2. Estadística i descripció

Aquesta vista està formada per un conjunt de caixes de text (*UILabels*) i una imatge (*UIImage*). Aquesta imatge s'obtindrà a partir d'una URL[102] que el servidor haurà generat utilitzant l'API de Google Charts. Aquesta imatge mostrarà un gràfic amb la informació de l'estadística seleccionada. Al prémer aquesta imatge es saltarà a una altre vista on es podrà veure amb més detall el

gràfic, podent-lo ampliar i moure a gust de l'usuari a través de varis gestos a la pantalla: un dit per desplaçar la imatge, dues picades per ampliar de cop i dos dits ampliar o reduir la imatge.

```
- (void)handleDoubleTap:(UIGestureRecognizer *)gestureRecognizer {
    float newScale = [imageScrollView zoomScale] * ZOOM_STEP;
    CGRect zoomRect = [self zoomRectForScale:newScale withCenter:[gestureRecognizer
        locationInView:gestureRecognizer.view]];
    [imageScrollView zoomToRect:zoomRect animated:YES];
}

- (void)handleTwoFingerTap:(UIGestureRecognizer *)gestureRecognizer {
    float newScale = [imageScrollView zoomScale] / ZOOM_STEP;
    CGRect zoomRect = [self zoomRectForScale:newScale withCenter:[gestureRecognizer
        locationInView:gestureRecognizer.view]];
    [imageScrollView zoomToRect:zoomRect animated:YES];
}
```

Fig. 5.94 Funcions per cada gest.

5.11. Implementació del mòdul Opcions

En aquest apartat s'explicarà el funcionament del mòdul Opcions, i també es veuran les diferents implementacions segons el sistema operatiu de l'aplicació (Android i iOS).

5.11.1. Funcionament

A continuació es mostra el diagrama amb el funcionament de les vistes de l'apartat d'Opcions i la seva interacció amb el servidor "Sóc de l'EETAC" i les APIs de LinkedIn i Twitter.

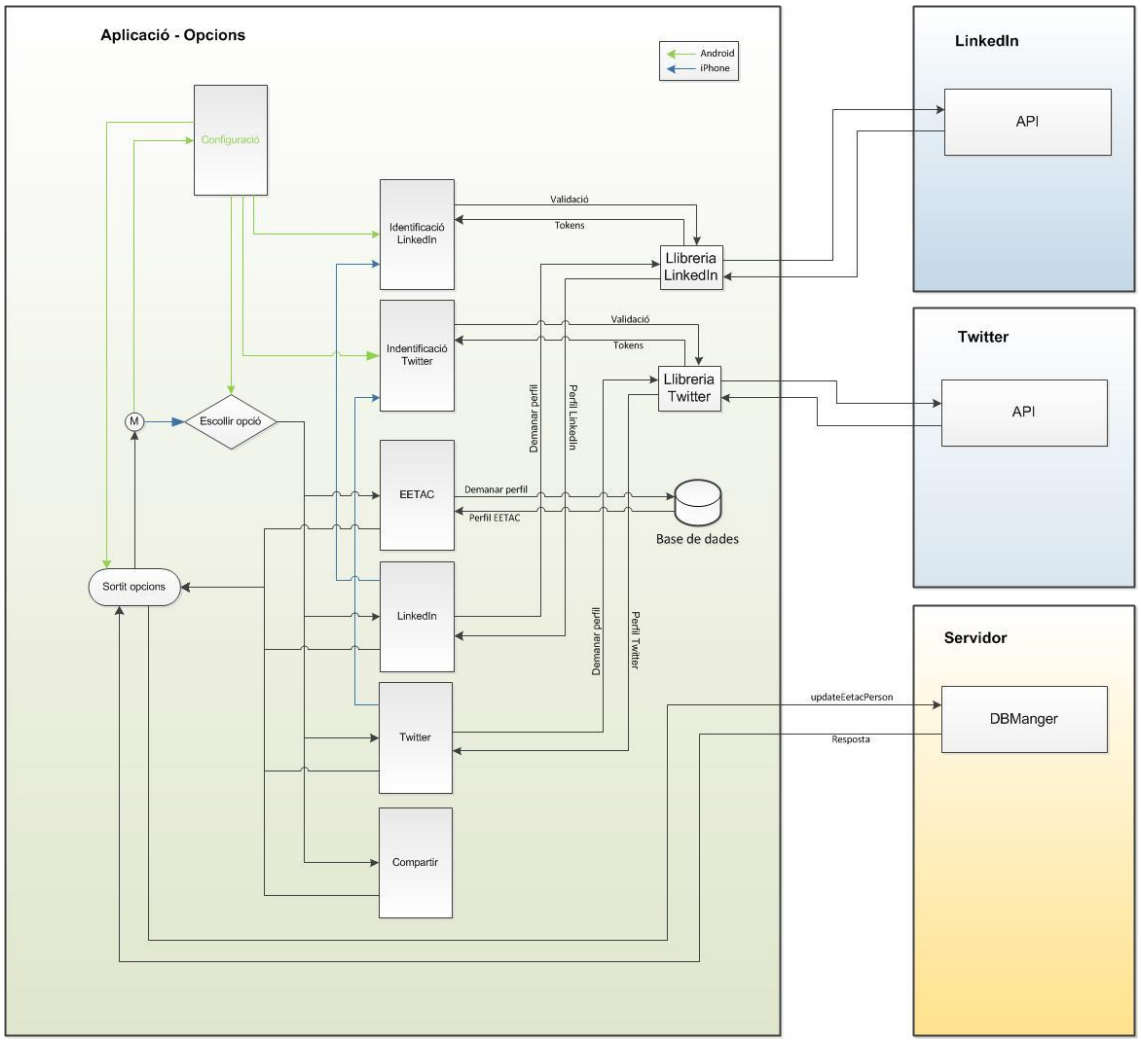

Fig. 5.95 Diagrama de funcionament de l'apartat Opcions

A l'apartat d'Opcions l'usuari pot definir quin sons els elements que desitja que siguin visibles per la resta d'usuaris de l'aplicació o si vol enviar o rebre missatges. També és la zona en que pot visualitzar la informació personal procedent de diferents fonts: EETAC, LinkedIn i Twitter. Una altre de les funcionalitats del apartat és de vincular i desvincular l'aplicació a les xarxes socials de LinkedIn i Twitter. I per acabar, i de forma transparent, l'usuari al sortir de l'apartat d'Opcions, actualitza de forma automàtica la informació al servidor de l'Escola.

5.11.2. Android

A continuació es presentaran les implementacions de la interfície gràfica Configuració, EETAC, LinkedIn, Twitter, Compartir, i Identificació LinkedIn i Twitter, presentats anteriorment a la **figura 5.95**.

Vegeu la **Fig. I.26** Menú de configuració, Android, la **Fig. I.27** Informació d'usuari procedent de l'EETAC, Android i iOS respectivament, la **Fig. I.28** Informació d'usuari procedent de LinkedIn, Android i iOS respectivament, la **Fig. I.29** Informació d'usuari procedent de Twitter, Android i iOS respectivament, i la **Fig. I.30** Vista per compartir la informació, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

5.11.2.1. Configuració

En el apartat de configuració trobem els diferents apartats que possibiliten al usuari realitzar modificacions sobre la informació que es disposa sobre ell.

Aquesta vista és diferent a totes les demes, ja que Android [66] permet la creació per defecte d'una vista de configuració. Un altre dels avantatges d'aquesta opció és que no s'ha de preocupar pel emmagatzematge i recuperació de dades ja que s'encarrega d'això cada vegada que s'obre la vista de configuració. Tanmateix es pot seguir tenint accés a aquestes dades i gestionar aquesta pantalla com una activitat més encara que es realitzi d'aquesta manera.

Cada vegada que es surt d'aquesta vista l'aplicació s'encarrega de connectar amb el servidor per actualitzar els possible canvis que s'hagin pogut produir. El que fa és en primer lloc carrega de la base de dades la informació que es té i es guarda en un objecte *EetacPerson* anomenat eetacShare. Tot seguit es canvia els estats del tots els *Allows* agafant la informació que el usuari tingui configurada, un cop modificats els *Allows* es torna a guarda la informació a la base de dades i s'envia al servidor.

```
eetacShare.setAllowCurrentJobs(checkboxPrefActualJobShare.isChecked());
eetacShare.setAllowImageLinkedIn(checkboxPrefImageShare.isChecked());
eetacShare.setAllowLocation(checkboxPrefPosition.isChecked());
eetacShare.setAllowName(checkboxPrefNameShare.isChecked());
eetacShare.setAllowSurname(checkboxPrefSurnameShare.isChecked());
eetacShare.setAllowPastJobs(checkboxPrefPastJobs.isChecked());
eetacShare.setAllowTitulacio(checkboxPrefTitulacioShare.isChecked());
eetacShare.setAllowEmailEstudiantEetac(checkboxeetacemail.isChecked());
eetacShare.setAllowDataTitulacio(checkboxenddate.isChecked());
eetacShare.setAllowMessages(checkboxsendrevicve.isChecked());
```

Fig. 5.96 Obtenint les dades de les preferències abans d'enviar-les al servidor.

5.11.2.2. EETAC

Aquesta vista mostra informació personal del usuari procedent de l'Escola, la carrega directament de la base de dades interna, ja que la descarrega s'ha fet

prèviament al iniciar l'aplicació. de moment la imatge que mostra procedeix de LinkedIn, tot i que en un futur es pretén que sigui la imatge que l'Escola té del usuari, aquesta imatge l'hauria de proporcionar el Bus SOA al servidor. És una vista molt semblant a la que es mostra al prémer un globus a la vista de Companys.

5.11.2.3. LinkedIn

En aquesta vista tota la informació que es visualitza prové del LinkedIn, i a part de la utilitat de mostra informació també fa la funció d'actualitzar la base de dades, ja que al entrar-hi l'aplicació es connecta de nou amb la xarxa social i obtenint la informació actualitzada, en cas que l'usuari l'hagi actualitzat.

5.11.2.4. Twitter

Aquesta vista nomes estarà activada en el cas que l'usuari hagi donat permisos a l'aplicació, en cas contrari el boto romandrà desactivat. La seva funció és de la de connectar-se al Twitter i mostra informació bàsica de la comte del usuari, com són la imatge personal, el nom d'usuari, el numero de tuits, de seguidors i el numero de comptes de Twitter que segueix.

5.11.2.5. Compartir

En el apartat compartit és on l'usuari podrà decidir quina és la informació que vol que els altres usuaris vegin sobre ell. La creació d'aquesta vista al contrari que les tres anteriors es crea de la mateixa manera que la vista de configuració. Cada vegada que es canvia algun dels cams automàticament és guarda en el objecte *eetacShare*, que més endavant al sortir de l'apartat de configuració serà guardat i enviat al servidor.

```
checkboxPrefPastJobs.setOnPreferenceChangeListener(new reference.OnPreferenceChangeListener()
{
        public boolean onPreferenceChange(Preference preference, Object newValue) {
                if (newValue.toString().equals("true")) {
                        eetacShare.setAllowPastJobs(true);
                } else {
                        eetacShare.setAllowPastJobs(false);
                }
                return true;
        }
});
```

Fig. 5.97 Codi de modificació del *AllowPastJob*.

5.11.2.6. Identificació Twitter

A qui és on podrem donar-li permisos a l'aplicació perquè utilitzi les funcionalitats del Twitter. Al prémer sobre aquesta opció l'aplicació ens redirigirà cap a la web del Twitter on l'usuari introduirà les seves credencials, el procés és pràcticament igual que al seguit per accedir al LinkedIn. Si pel contrari es desmarca aquesta opció l'aplicació elimina les claus que guardades.

```
checkboxPreftw.setOnPreferenceChangeListener(new Preference.OnPreferenceChangeListener() {
public boolean onPreferenceChange(Preference preference, Object newValue) {
        if (newValue.toString().equals("true")) {
                confTwitter.setEnabled(true);
                Intent i = new Intent(getApplicationContext(), TwitterActivity.class);
                startActivity(i);
        } else {
                credentials = new Credentials(sharedPrefs);
                credentials.deleteTw();
                confTwitter.setEnabled(false);
                }
        return true;}});
```

Fig. 5.98 Codi de modificació per a la casella de Twitter.

5.11.2.7. Identificació LinkedIn

Aquesta casella serveix al igual que l'anterior per vincular o desvincular l'aplicació amb el LinkedIn, amb la diferencia que si no s'entra com a convidat estarà marcada per defecte. Al desmarcar la casella l'aplicació passarà al mode de convidat automàticament i tant l'apartat de Companys com el de Missatges es veuran deshabilitats.

5.11.3. iOS

A continuació es presentaran les implementacions de la interfície gràfica Configuració, EETAC, LinkedIn, Twitter i Compartir, presentats anteriorment a la **figura 5.95**.

Vegeu la **Fig. I.26** Menú de configuració, Android, la **Fig. I.27** Informació d'usuari procedent de l'EETAC, Android i iOS respectivament, la **Fig. I.28** Informació d'usuari procedent de LinkedIn, Android i iOS respectivament, la **Fig. I.29** Informació d'usuari procedent de Twitter, Android i iOS respectivament, i la **Fig. I.30** Vista per compartir la informació, Android i iOS respectivament, per observar l'aspecte de la interfície gràfica un cop implementada.

De la mateixa manera que a l'apartat de Missatges, les següent quatre vistes en realitat només són una, i aquesta està composta per una taula (*UITableView*) i per una barra (*UIBottomBar*) a la part inferior de la vista on hi ha les pestanyes o segments (*UISegmentedControl*).

5.11.3.1. EETAC

Si s'està validat a l'aplicació a través de LinkedIn el primer que realitza la vista a través de la funció *viewDidLoad* és connectar-se a la base de dades de l'aplicació utilitzant la llibreria Core Data i obtenir les dades de l'usuari i mostrar-la a través de diferents caixes de text (UILabels) dins d'una cel·la (*UITableViewCell*) de la taula. Aquesta vista mostra el nom i cognoms, la imatge, la titulació i la data d'obtenció de la titulació de l'estudiant.

5.11.3.2. LinkedIn

En el cas que l'usuari entri com a convidat no es realitza cap connexió i s'accedeix directament a la pestanya de *LinkedIn*, anul·lant l'accés a les pestanyes de l'EETAC i compartir. En aquesta situació aquesta vista només et permet i mostra un botó (*UIButton*) validar-te a través de LinkedIn de la mateixa forma que es realitza a la vista inicial d'Accés a l'aplicació.

En cas contrari, al prémer aquesta pestanya la vista realitzarà una connexió a l'API de LinkedIn a través de la seva llibreria, per refrescar la informació de l'usuari. Aquesta informació es mostrarà de la mateixa manera a través de caixes de text dins de les cel·les. La informació es mostrarà per seccions, a la primera hi haurà les dades personals: imatge, noms i cognoms del perfil de LinkedIn i l'estat de la connexió, també disposarem d'un botó (*UIButton*) per desvincular la nostre compte de l'aplicació; la segona i tercera secció mostraran el llistat feines actuals i passades respectivament, també a través de diferents caixes de text dins de cel·la. En aquest cas un el contingut d'una feina es mostra en una sola cel·la.

5.11.3.3. Twitter

En el cas que no s'hagi vinculat la compte de Twitter a l'aplicació, aquesta pestanya mostrà un botó (*UIButton*) que permetrà anar a la Configuració de l'iOS [86] per enllaçar la nostra compte de Twitter.

En cas contrari, l'aplicació accedirà a la compte de Twitter de l'usuari al sistema d'iOS i a través de la llibreria de Twitter es mostrarà la imatge i el nom d'usuari, l'estat, el nombre de tuits, amics i seguidors de Twitter; tot a través de diferents caixes de text dins de les cel·les de la taula.

5.11.3.4. Compartir

Aquesta vista està formada deu cel·les fixes, a l'interior de cada cel·la, a la part esquerra es mostra una caixa de text (*UILabel*) que ens informa de quin permís es cedirà a l'aplicació i a la part dreta tenim un interruptor (*UISwitch*) per activar o desactivar aquest permís.

Un cop s'hagi configurat totes les opcions a compartir s'han de guardar totes aquestes preferències. Això es realitza quan es surt de l'apartat d'opcions. Aquesta acció es realitza a través de la funció *viewWillDisappear*. En aquesta funció es realitzen dues operacions importants en el cas d'estar validat a l'aplicació:

- Es realitza una connexió a Core Data per actualitzar tota la informació de l'usuari, tant la de LinkedIn, com a la de Twitter, com les opcions de compartir informació amb altres usuaris.
- Es realitza una connexió al servidor de l'EETAC, pel el mateix motiu anterior, actualitzar tota la informació i preferències de l'usuari al servidor. Aquesta connexió es realitza a través de la URL [102] *updateEetacPerson*.

```
- (void)viewWillDisappear:(BOOL)animated {
    if (!linkedInLoginTapped && ![me.eetacID isEqualToString:@""]) {
        me.allowAcces = allowAcces;
        me.allowName = [[allowedOptionsDetail objectAtIndex:0] boolValue];
        me.allowSurname = [[allowedOptionsDetail objectAtIndex:1] boolValue];
        me.allowUrlPicture = [[allowedOptionsDetail objectAtIndex:2] boolValue];
        me.allowEmail = [[allowedOptionsDetail objectAtIndex:3] boolValue];
        me.allowLocation = [[allowedOptionsDetail objectAtIndex:4] boolValue];
        me.allowMessages = [[allowedOptionsDetail objectAtIndex:5] boolValue];
        me.allowDegree = [[allowedOptionsDetail objectAtIndex:6] boolValue];
        me.allowDegreeDate = [[allowedOptionsDetail objectAtIndex:7] boolValue];
        me.allowCurrentJobs = [[allowedOptionsDetail objectAtIndex:8] boolValue];
        me.allowPastJobs = [[allowedOptionsDetail objectAtIndex:9] boolValue];

        if (self.engine.isAuthorized ) {
            [ObjectMeCoreData updateMeObjectWithEetacPerson:me
                WithContext:managedObjectContext];
            [ObjectMeCoreData printEetacPersonWithContext:managedObjectContext];
            me.linkedinID = linkedInPerson.linkedinID;
            me.name = linkedInPerson.name;
            me.surname = linkedInPerson.surname;
            me.urlPicture = linkedInPerson.urlPicture;
            me.currentJobs = currentJobs;
            me.pastJobs = pastJobs;
        }
        else {
            [ObjectMeCoreData deleteMeObjectWithContext:managedObjectContext];
        }
        JSON *json = [[JSON alloc] init];
        [json setJsonDelegate:self];
        [json postAsyncronousJsonFromNSDictionary:[me
                getEetacPersonDictionaryFromEetacPerson:me] withPath:@"updateEetacPerson"];
    }
}
```

Fig. 5.99 Actualitzar la informació de l'usuari tant a la base de dades interna de l'aplicació, com a la base de dades del servidor

En el cas que l'usuari de l'aplicació hagi accedit com a convidat, al sortir de l'apartat d'Opcions, no es realitzarà cap connexió.

CAPÍTOL 6. PLANIFICACIÓ I DESENVOLUPAMENT

Per a l'èxit d'un projecte és necessari fer una bona planificació de temps. Aquesta planificació servirà per analitzar el progrés del projecte i per saber si és possible aconseguir els objectius proposats en un termini definit.

En aquest capítol veurem com s'ha planificat aquest projecte i mostrarem les fases de disseny i desenvolupament del mateix. També cal esmentar que en aquest projecte inicialment estaven involucrats dos estudiants més, ja que, tal i com s'ha vist en els primers capítols, la idea del projecte donava moltes possibilitats.

6.1. Disseny

En aquesta fase, un grup de quatre estudiants i els directors del projecte van realitzar un seguit de reunions quinzenals on conjuntament definíem els requisits, es proposaven idees, i es dissenyava el cos del projecte global.

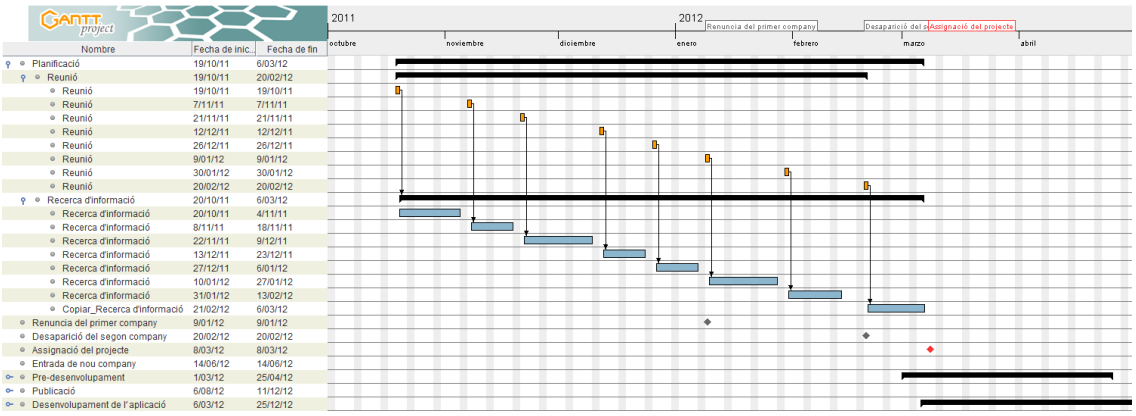

Fig. 6.1 Diagrama de Gantt, fase de disseny

6.2. Assignació de tasques

Un cop realitzada la fase de disseny, es van assignar les tasques que havia de realitzar cada un dels membres del projecte. En aquell moment dos membres del grup van abandonar el projecte i es van haver de reajustar les tasques assignades. Es va passar a l'aprenentatge dels coneixements necessaris perquè cadascú pogués realitzar les seves tasques i començar la presa de contacte amb les diferents eines que ens han acompanyat durant tot el desenvolupament del projecte, l'Eclipse [77] en el cas de l'aplicació per Android [66] i l'Xcode [103] en el cas de l'aplicació per iPhone. Al final d'aquesta fase es

va fer l'assignació formal del TFC, en aquest moments ja teníem ben definits els objectius i quina seria la funció de cada membre del grup.

6.3. Desenvolupament

En aquesta fase, es comença el desenvolupament de les aplicacions. El ritme de reunions va disminuint ja que ja es té clar el que es vol aconseguir i només queda el treball de desenvolupament. A mesura que va passant el temps es van implementant els diferents apartats, i a la vegada també es treballa amb la part de servidor.

A la recta final del projecte, es combina el desenvolupament de l'aplicació amb l'elaboració de la memòria, fins que finalment es dóna per acabada l'aplicació i es concentra tot el treball únicament a la creació de la memòria.

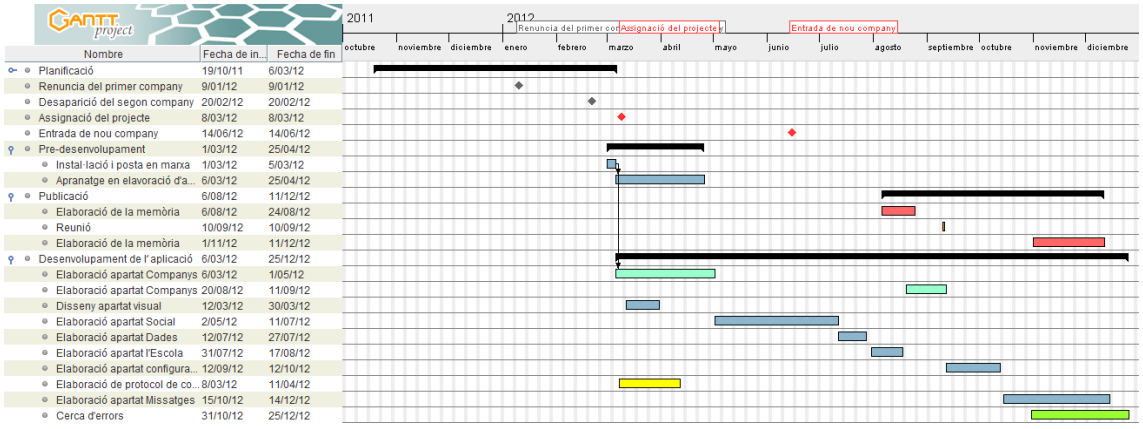

Fig. 6.2 Diagrama de Gantt, fase de desenvolupament

6.4. Presentació

En aquesta fase, i amb l'avantatge d'haver realitzat el projecte conjuntament, realitzarem una demostració pràctica de l'aplicació "Sóc de l'EETAC", presentant al mateix temps l'aplicació desenvolupada per Android i iOS i com interaccionen entre elles i el servidor.

CAPÍTOL 7. CONCLUSIONS

7.1. Resultats

El resultat d'aquest projecte ha estat el desenvolupament de l'aplicació "Sóc de l'EETAC" tant per a telèfons Android com per iPhone. S'han complert tots els requeriments i gran part dels objectius proposats.

El primer objectiu era facilitar la recollida d'informació laboral dels antics alumnes de l'EETAC a través de la xarxa social de LinkedIn. La vinculació de l'aplicació a la xarxa professional de LinkedIn ens ha proporcionat una molt bona font d'informació sobre els titulats/des de l'EETAC i també ens ha servit com a mètode per autentificar i validar aquest usuaris a la base de dades d'estudiants de l'Escola.

El segon objectiu era facilitar la comunicació dels antics alumnes a través d'un servei de missatges propi de l'aplicació. Amb la implementació d'un servei de missatges propi juntament amb la xarxa social de Twitter s'ha complert amb aquest objectiu donant dues alternatives de comunicació.

El tercer objectiu era geoposicionar [78] els usuaris de l'aplicació i els seus llocs de treball. Al mòdul Companys, l'aplicació permet posicionar els usuaris, però a l'hora de posicionar les empreses, s'ha realitzat una implementació parcial, ja que l'obtenció de les dades de localització no es pot fer directament de la xarxa professional LinkedIn.

El quart objectiu era mostrar els continguts de les xarxes socials que utilitza l'Escola: LinkedIn, Twitter i interactuar amb ells. Al mòdul Social, no només s'ha complert l'objectiu, sinó que s'ha ampliat mostrant les xarxes socials de Facebook, Google+ i YouTube, i també donant la possibilitat d'interactuar amb alguna d'elles, fent als usuaris més participatius.

El cinquè objectiu era visualitzar tot un seguit de dades i estadístiques referents a l'Escola per promocionar-la. Tot i que al mòdul Dades, de moment no proporciona informació real, l'aplicació està preparada per mostrar les dades i estadístiques que l'Escola vulgui transmetre als usuaris de l'aplicació.

Per acabar, l'últim objectiu era connectar-se i intercanviar dades amb el servidor del sistema "Sóc de l'EETAC". Tot aquest flux de dades s'obté de diverses fonts i s'ha implementat un servidor on ens connectem contínuament per enviar i rebre posicions, missatges, estadístiques i pàgines webs d'interès, etc. Aquestes peticions també s'han hagut de definir, dissenyar i implementar-les al servidor, creant així una API per al nostre TFC . A part, la connexió també es realitza amb els servidors de LinkedIn i Twitter, a través de les seves APIs.

Tot i que el propòsit del projecte era destinat a titulats i titulades de l'EETAC, al llarg del seu desenvolupament i implementació, ens hem adonat que les aplicacions mòbils podrien anar destinades als actuals estudiant i també als

futurs, com a eina de de promoció de l'Escola. Per aquest motiu també s'han incorporat referencies als serveis que els actuals estudiants utilitzen.

7.2. Millores futures

En el món del desenvolupament de software, podríem dir que una aplicació mai esta finalitzada del tot, això provoca que l'aplicació sempre tingui opcions de millora, des de solucionar problemes imprevistos pels desenvolupadors (nosaltres mateixos), a afegir noves funcionalitats que facin l'aplicació més atractiva i útil per al usuari final.

A continuació es mostrà tot un seguit de possibles millores que es podrien implementar a cada mòdul de l'aplicació i unes altres millores orientades a àmbits més generals de l'aplicació.

7.2.1. Accés a l'aplicació

Actualment la validació d'un usuari es realitza a través de la xarxa professional de LinkedIn. Una millora seria incorporar el sistema de validació propi de l'UPC, així els estudiants podrien accedir a l'aplicació amb el seu usuari i contrasenya habituals de l'Escola (Atenea, Netàrea, correu electrònic...).

7.2.2. Menú

El menú és la primera vista que ens mostra l'aplicació i conté els botons amb els diferents mòduls i a partir d'aquí ens podem moure a través d'ella. Una millora seria reconvertir aquesta primera vista en una espècie de resum, notes, avisos i notícies personalitzades per cada usuari i que l'accés als mòduls de l'aplicació es realitzés des d'un menú lateral ocult i desplegable, aquest menú desplegable es podria implementar-se a cada vista facilitant molt més la navegació a través de tota l'aplicació.

7.2.3. Mòdul L'Escola

En aquesta mòdul es pot accedir a varis serveis la universitat. Però utilitzant l'autentificació de l'UPC a l'inici de l'aplicació es podria implementar un servei natiu per Android i iOS de les intranets de la universitat: Atenea, Netàrea, correu electrònic, etc. D'aquesta manera l'aplicació seria una eina més completa.

7.2.4. Mòdul Missatges

Aquest mòdul té implementat un sistema de missatges propi i un altre a través de Twitter. Els dos sistemes s'haurien de millorar incorporant un sistema d'avisos a l'usuari per l'arribada de nous missatges, ja estigui dins de l'aplicació o fora d'ella. També es podria millorar el servei indicant a través de globus o algun altre mètode quins són els nous missatges, quins s'han llegit i quins no. Una altre millora seria canviar el sistema actual orientat a missatges enviats i rebuts a un sistema orientat a conversa o xat. Pel sistema de missatgeria a través de Twitter una possible millora seria implementar l'enviament de missatges privats a les comptes de Twitter, ja que ara els missatges són públics i només es filtren per un *hashtag [82]*.

7.2.5. Mòdul Companys

En aquest mòdul es mostra la localització dels usuaris de l'aplicació, queda per millorar la implementació per mostrar la localització de les empreses on treballen els usuaris, de manera que es pugui diferenciar on es troba una persona ara mateix i on està localitzat el seu lloc de treball. També es podria millorar aquesta implementació afegint la localització d'altres empreses que proporcionessin ofertes de treball. En aquest mòdul també seria necessari afegir un llistat amb els usuaris de l'aplicació, per facilitar la localització d'aquest i seguint aquesta idea també és podria millorar la visualització de les xinxetes de manera que fos més senzill identificar un usuari al mapa.

7.2.6. Mòdul Social

En aquest mòdul es mostren les xarxes socials on l'Escola està present. La xarxa de LinkedIn i Twitter tenen una implementació a través de les seves respectives APIs [68]. Una possible millora seria implementar la interfície gràfica a través de les APIs de les restants xarxes socials de l'aplicació: Facebook, Google+ i YouTube. També es podria millorar la integració de totes les xarxes socials amb l'aplicació i fomentar la interacció amb aquestes xarxes socials des de qualsevol punt de l'aplicació, per exemple, agregant empreses a LinkedIn, afegir amics, comentar a Facebook, etc.

7.2.7. Mòdul Dades

En aquest mòdul, les gràfiques actuals són imatges generades a partir de l'API de Google Charts i enviada directament a l'aplicació des del servidor que s'ha implementat. La millora podria ser un canvi de mètode i rebre només la informació de les dades i generar des de l'aplicació les gràfiques necessàries.

D'aquesta manera podrien ser interactives per l'usuari, permetent-li escollir el rang de les dades, ampliar o reduir escala, canviar el tipus de gràfica, etc.

7.2.8. Mòdul Opcions

Si s'implementen algunes de les millores esmentades anteriorment s'hauria d'ampliar el mòdul d'opcions afegint els accessos i validacions a les noves xarxes socials o ampliant la varietat de permisos per compartir dades amb altres usuaris de l'aplicació.

7.2.9. Altres millores

7.2.9.1. Ofertes de treball

Una altre millora podria ser la creació d'un mòdul nou on les empreses i la universitat poguessin mostrar ofertes de treball o pràctiques d'empreses i que l'usuari pogués interactuar amb elles, apuntant-se a aquestes ofertes, o localitzar-les en el mapa, o la possibilitat de contactar directament amb aquestes empreses, etc.

7.2.9.2. Seguretat

Les connexions entre l'aplicació i el servidor, actualment es realitzen a través d'HTTP [84], una millora seria implementar el mateix servei REST [98] que s'utilitza ara però utilitza l'HTTP segur (HTTPS), per tal d'evitar possibles suplantacions d'identitat en el sistema. També es podrien buscar altres mètodes per millorar la seguretat de l'aplicació.

7.2.9.3. Interactivitat

Com ja s'ha comentat anteriorment es podria millorar la interactivitat de l'aplicació a través de les xarxes socials, però també es podria millorar l'accés a diferents punts de l'aplicació des d'altres, de manera que l'usuari hagués de fer menys passos per navegar per l'aplicació, ja sigui a través del menú desplegable comentat anteriorment o a través de botons dins de la vista, com per exemple el que ja tenim dins del mòdul Companys que permet enviar un missatge a un usuari sense tenir que accedir al mòdul de Missatges.

7.2.9.4. Adaptar l'aplicació a Tablets

Tant l'aplicació per desenvolupada per Android i iOS [86] es poden executar a les *Tablets* del mateix sistema operatiu respectivament. Una possible millora seria redissenyar la interfície gràfica i adaptar l'aplicació a les *Tablets* ja que disposen de major espai de treball que un telèfon mòbil.

7.2.9.5. Adaptar l'aplicació a diferents idiomes

Tant l'aplicació desenvolupada per Android com la desenvolupada per iOS poden suportar varis idiomes, però cal implementar-los.

7.3. Impacte mediambiental

En aquests projecte intervenen dos tipus d'aparells, el telèfon mòbil i el servidor, el seu impacte mediambiental en la realització del projecte no va més enllà del seu consum energètic.

En el cas del servidor el programa que s'executa ja ho fa en una màquina virtual, d'aquesta manera es consumeixen menys recursos i energia que si ho féssim amb una màquina dedicada, ja que una sola màquina pot contenir més d'una màquina virtual.

En el cas de l'aplicació pel telèfon mòbil, un usuari podria començar a canviar els seus hàbits i utilitzar habitualment l'aplicació mòbil de l'Escola ja que disposa dels mateixos serveis, que un ordinador. En aquest cas el consum energètic d'un ordinador seria més elevat que el d'un telèfon mòbil i aconseguiríem reduir l'impacte mediambiental.

CAPÍTOL 8. BIBLIOGRAFIA

8.1. Referències Android

[1] Curs Android: http://www.sgoliver.net

[2] Guia de referència per a desenvolupadors Android: http://www.wikidroid.es

[3] Snippets (trossos de codi): http://es.wikicode.org

[4] Tutorial REST : http://timewasted.net

[5] Tutorial per a mapes: http://jonsegador.com

[6] API Android: http://developer.android.com/

[7] Tutorial per mòdul Menú: http://www.androidhive.info

[8] Tutorial de la llibreria GreenDroid: http://androcode.es/

[9] Tutorials REST i llibreria Gson: http://www.negomobile.es/es/android

[10] Guia de Gson: https://sites.google.com/site/Gson/Gson-user-guide

[11] Tutorial de Twitter: http://androideity.com/

8.2. Referències iOS

[12] Bennett, G. Fisher, F. Lees, L. *Objective-C for Absolute Beginners (iPhone, iPad, and Mac Programming Made Easy)*, Apress® (2010). ISBN-13 (pbk): 978-1-4302-2832-5.

[13] Mark, D. Nutting, J. LaMarche, J. *BeginningiPhone 4 Development (Exploring the iOS SDK)*, Apress® (2011). ISBN-13 (pbk): 978-1-4302-3024-3.

[14] Kochan, S.G. *Programming in Objective-C*. Addison-Wesley (2011). ISBN-13: 978-0-321-71139-7

[15] Tutorial JSON: http://www.hbensalem.com/iphone-2/iphone-synchronous-and-asynchronous-json-parse/

[16] Tutorial Twitter: http://mobile.tutsplus.com/tutorials/iphone/ios-sdk_twitter-framework_twrequest/

[17] Diversos tutorials iOS: http://www.raywenderlich.com/tutorials

[18] Tutorial cel·les: http://useyourloaf.com/blog/2010/10/04/swiping-to-delete-rows-from-a-table.html

[19] Importar llibreries estàtiques: http://blog.carbonfive.com/2011/04/04/using-open-source-static-libraries-in-xcode-4/

[20] Tutorial Core Data: http://timroadley.com/2012/02/19/core-data-basics-part-4-relationships/

[21] Tutorial Core Data: http://maniacdev.com/2012/03/tutorial-getting-started-with-core-data-in-ios-5-using-xcode-storyboards/

[22] Tutorial Map Kit: http://www.edumobile.org/iphone/iphone-programming-tutorials/mapkit-example-in-iphone/

8.3. Referències Servidor

[23] Tutorial JSON Apache Tomcat: http://www.vogella.com/

[24] Servei REST: http://jersey.java.net/nonav/documentation/latest/json.html

[25] Càlcul de distàncies entre coordenades: http://www.movable-type.co.uk/scripts/latlong-vincenty.html

[26] Implementació càlcul distàncies: http://stackoverflow.com/questions/120283/working-with-latitude-longitude-values-in-java

[27] Creació d'un hash: http://stackoverflow.com/questions/1209633/create-hash-from-string-and-int

[28] RFC 4627: http://www.ietf.org/rfc/rfc4627.txt

[29] Singleton: http://www.javaworld.com/javaworld/jw-04-2003/jw-0425-designpatterns.html

8.4. Referències Comunes

[30] API de Twitter: https://dev.twitter.com/docs

[31] API de LinkedIn: https://developer.linkedin.com/

[32] Fòrum de preguntes i respostes de programació: http://stackoverflow.com/

[33] Validador JSON: http://jsonlint.com

[34] Quota de mercat de telèfons intel·ligents: http://www.xatakamovil.com/sistemas-operativos/android-se-hace-con-el-75-del-mercado-en-el-tercer-trimestre-segun-idc

8.5. Xarxes socials i professionals de l'Escola

[35] LinkedIn EETAC Alumni: http://www.linkedin.com/groups/EETAC-Alumni-former-EPSC-EUPBL-86759

[36] Twitter: https://twitter.com/EETAC_UPC

[37] Facebook: https://www.facebook.com/EETAC

[38] Google+: http://plus.google.com/107196343161819110849

[39] YouTube: http://www.youtube.com/user/upceetac

Annex I. Interfície Gràfica

I.1. Índex de figures

Fig. I.1 Vista d'inici, iOS 112
Fig. I.2 Accés a l'aplicació, Android i iOS respectivament 112
Fig. I.3 Autentificació a LinkedIn, Android i iOS respectivament 113
Fig. I.4 Vista del Menú, Android i iOS respectivament 113
Fig. I.5 Llistat de pàgines web i serveis, Android i iOS respectivament 114
Fig. I.6 Exemple d'una pàgina web, Android i iOS respectivament 114
Fig. I.7 Llistat d'usuaris per enviar missatges, Android i iOS respectivament 115
Fig. I.8 Vista per enviar missatges, Android i iOS respectivament 115
Fig. I.9 Llistat de missatges enviats, Android i iOS respectivament 116
Fig. I.10 Vista d'un missatge enviat o rebut, Android i iOS respectivament ... 116
Fig. I.11 Llistat de tuits i mencions, Android 117
Fig. I.12 Llistat de tuits i mencions, iOS 117
Fig. I.13 Vista d'un tuit o menció, Android i iOS respectivament 118
Fig. I.14 Respondre una menció, Android i iOS respectivament 118
Fig. I.15 Mapa amb companys, Android i iOS respectivament 119
Fig. I.16 Informació d'un company, Android i iOS respectivament 119
Fig. I.17 Llistat de discussions de l'EETAC Alumni a LinkedIn, Android i iOS respectivament 120
Fig. I.18 Vista d'una discussió, Android i iOS respectivament 120
Fig. I.19 Llistat de tuits de L'Escola, Android i iOS respectivament 121
Fig. I.20 Vista d'un tuit, Android i iOS respectivament 121
Fig. I.21 Xarxa social Facebook, Android i iOS respectivament 122
Fig. I.22 Xarxa social Google+, Android i iOS respectivament 122
Fig. I.23 Xarxa de vídeos YouTube, Android i iOS respectivament 123
Fig. I.24 Llistat de dades i estadístiques, Android i iOS respectivament 123
Fig. I.25 Gràfica d'una estadística, Android i iOS respectivament 124
Fig. I.26 Menú de configuració, Android 124
Fig. I.27 Informació d'usuari procedent de l'EETAC, Android i iOS respectivament 125
Fig. I.28 Informació d'usuari procedent de LinkedIn, Android i iOS respectivament 125
Fig. I.29 Informació d'usuari procedent de Twitter, Android i iOS respectivament 126
Fig. I.30 Vista per compartir la informació, Android i iOS respectivament 126

I.2. Inici

Fig. I.1 Vista d'inici, iOS

I.3. Accés a l'aplicació

Fig. I.2 Accés a l'aplicació, Android i iOS respectivament

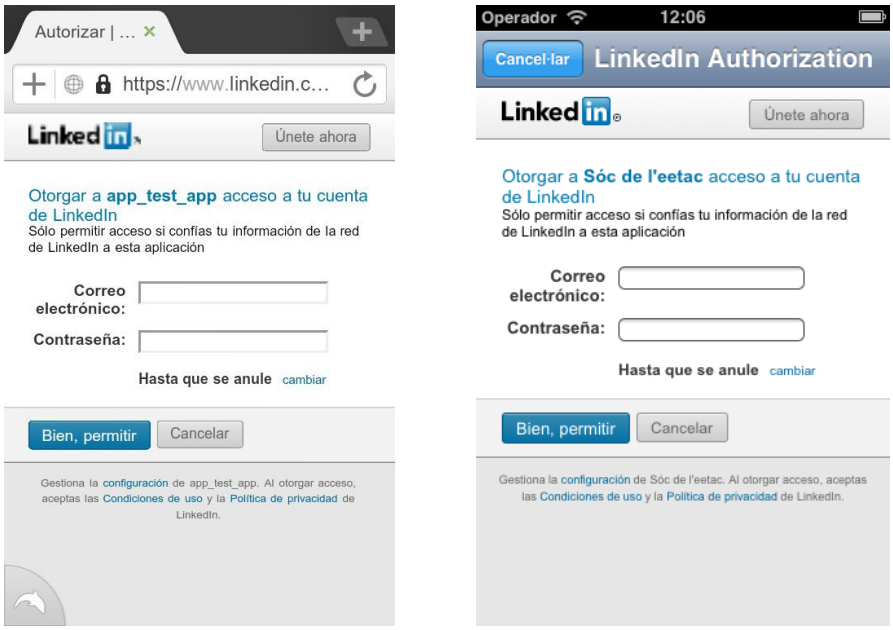

Fig. I.3 Autentificació a LinkedIn, Android i iOS respectivament

I.4. Menú

Fig. I.4 Vista del Menú, Android i iOS respectivament

I.5. Mòdul L'Escola

Fig. I.5 Llistat de pàgines web i serveis, Android i iOS respectivament

Fig. I.6 Exemple d'una pàgina web, Android i iOS respectivament

I.6. Mòdul de Missatges

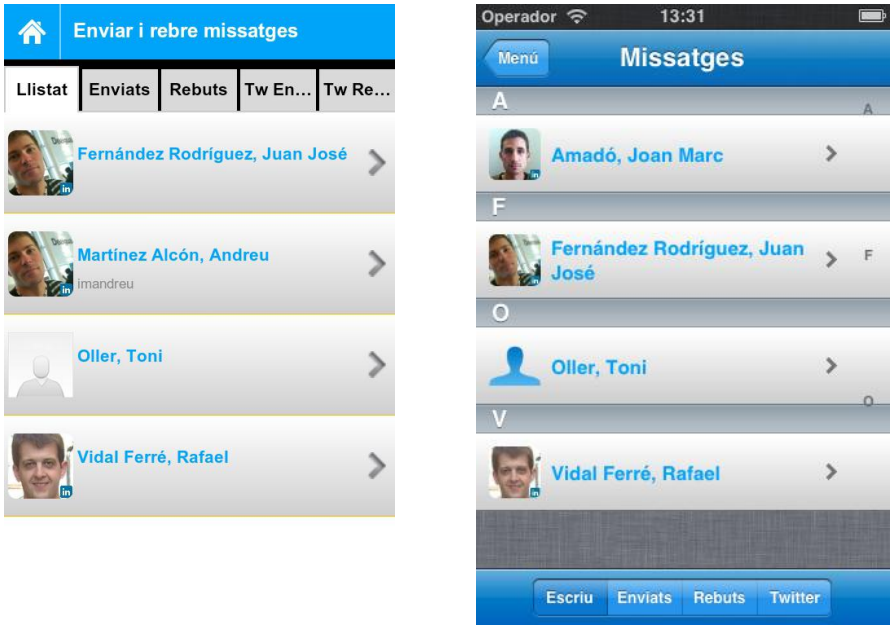

Fig. I.7 Llistat d'usuaris per enviar missatges, Android i iOS respectivament

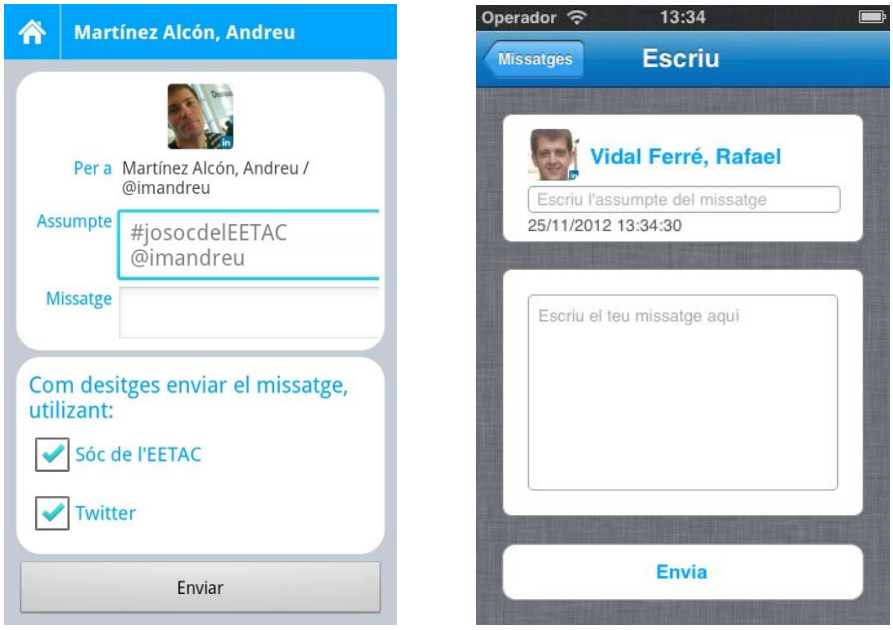

Fig. I.8 Vista per enviar missatges, Android i iOS respectivament

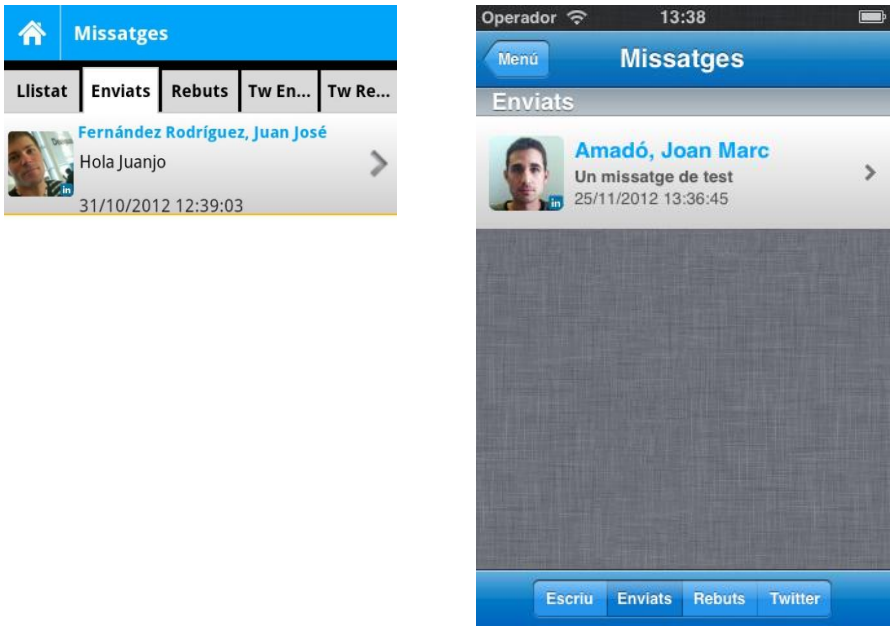

Fig. I.9 Llistat de missatges enviats, Android i iOS respectivament

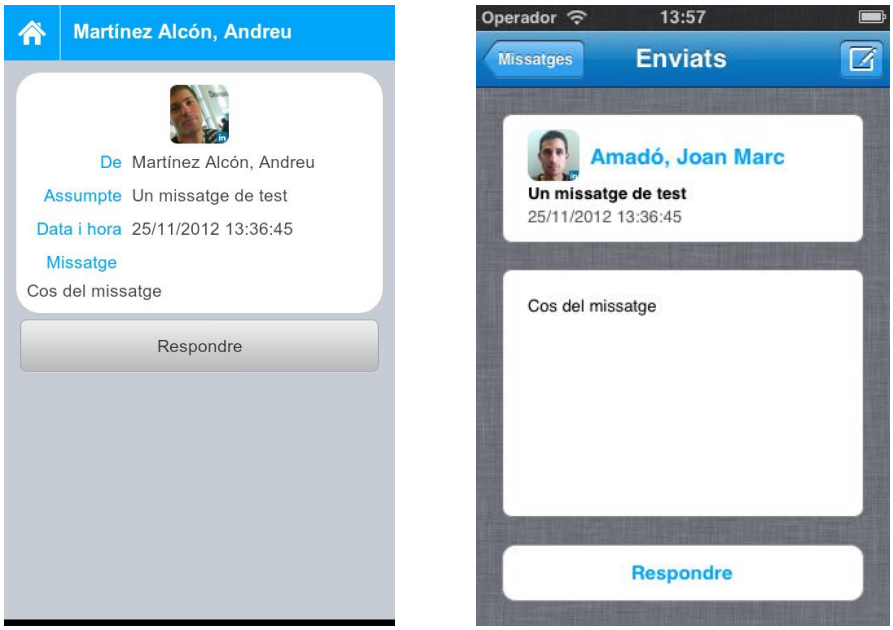

Fig. I.10 Vista d'un missatge enviat o rebut, Android i iOS respectivament

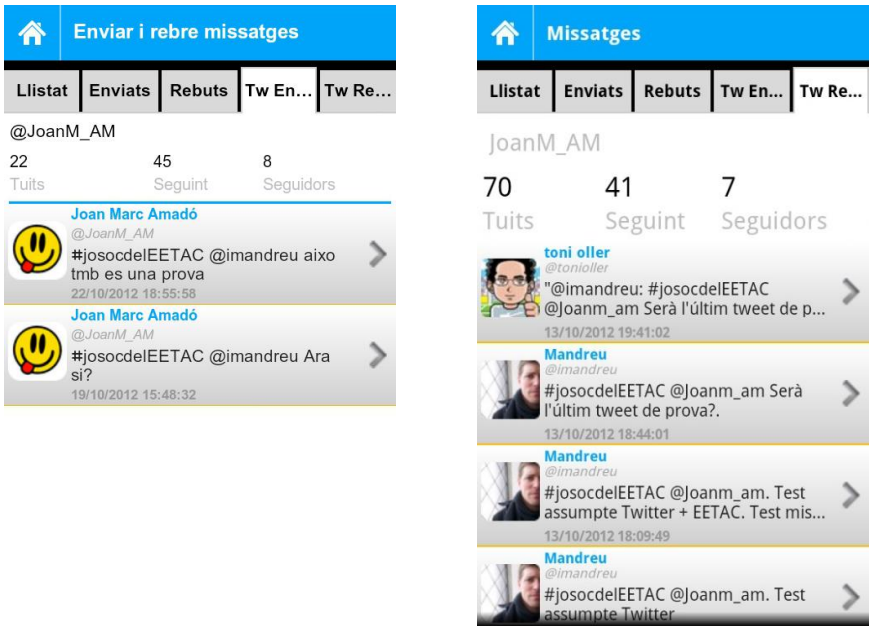

Fig. I.11 Llistat de tuits i mencions, Android

Fig. I.12 Llistat de tuits i mencions, iOS

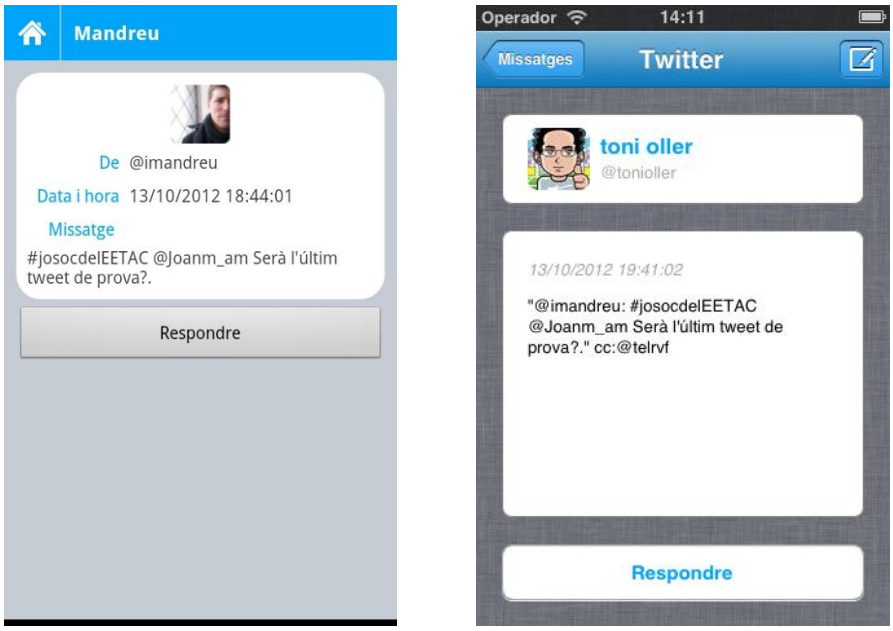

Fig. I.13 Vista d'un tuit o menció, Android i iOS respectivament

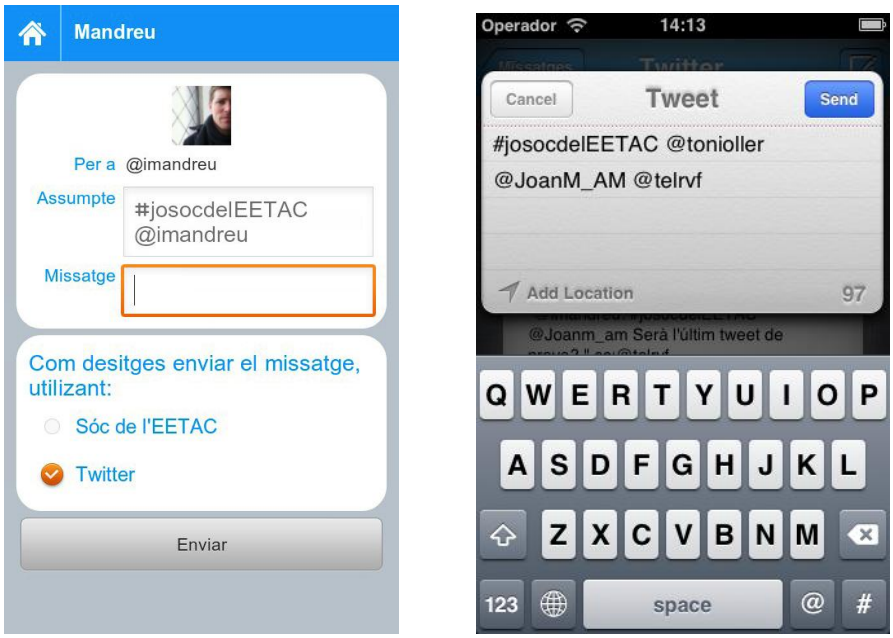

Fig. I.14 Respondre una menció, Android i iOS respectivament

I.7. Mòdul Companys

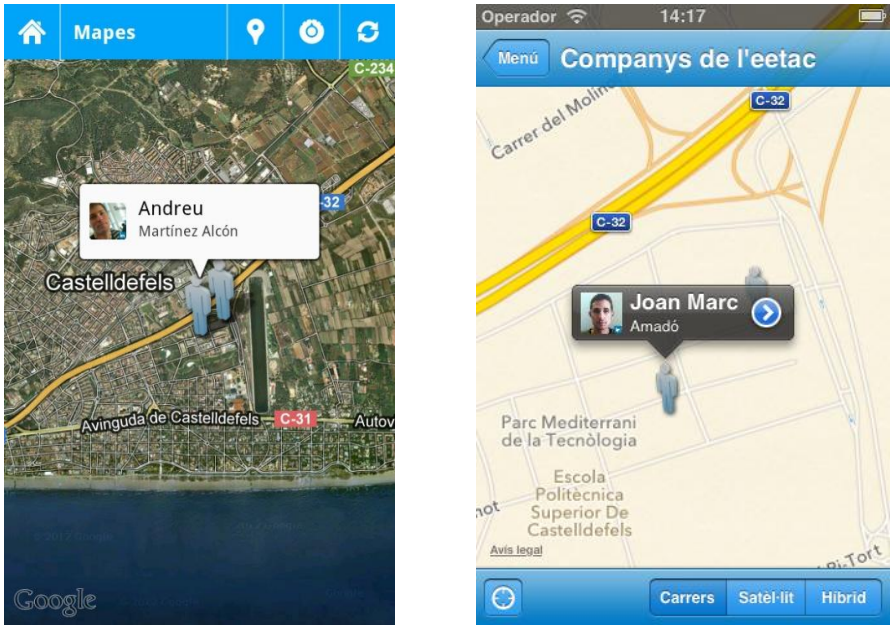

Fig. I.15 Mapa amb companys, Android i iOS respectivament

Fig. I.16 Informació d'un company, Android i iOS respectivament

I.8. Mòdul Social

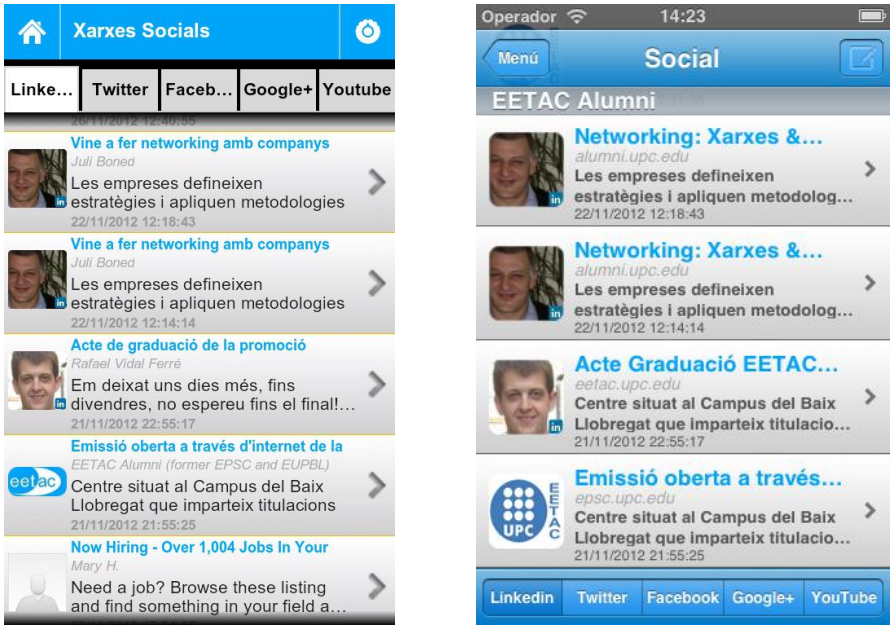

Fig. I.17 Llistat de discussions de l'EETAC Alumni a LinkedIn, Android i iOS respectivament

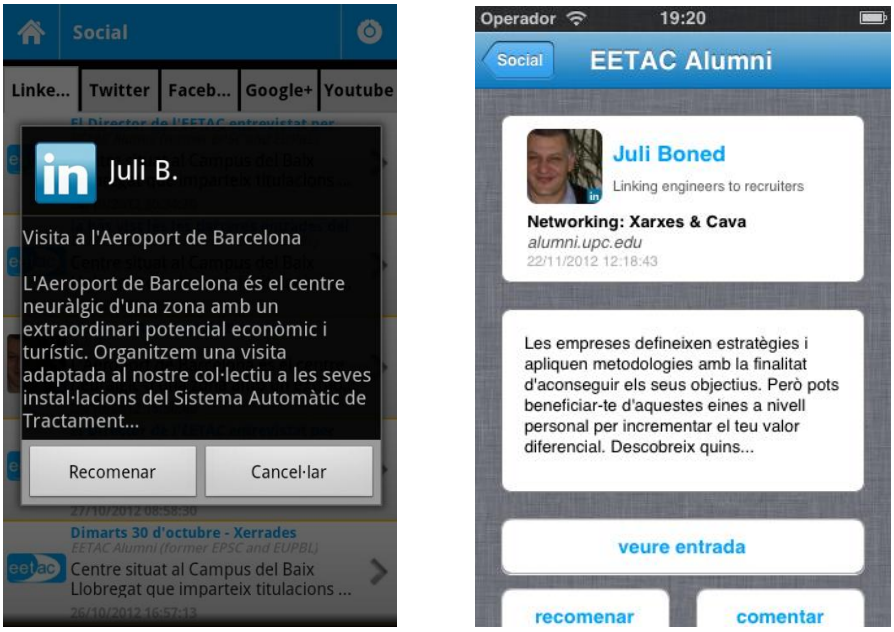

Fig. I.18 Vista d'una discussió, Android i iOS respectivament

Fig. I.19 Llistat de tuits de L'Escola, Android i iOS respectivament

Fig. I.20 Vista d'un tuit, Android i iOS respectivament

Fig. I.21 Xarxa social Facebook, Android i iOS respectivament

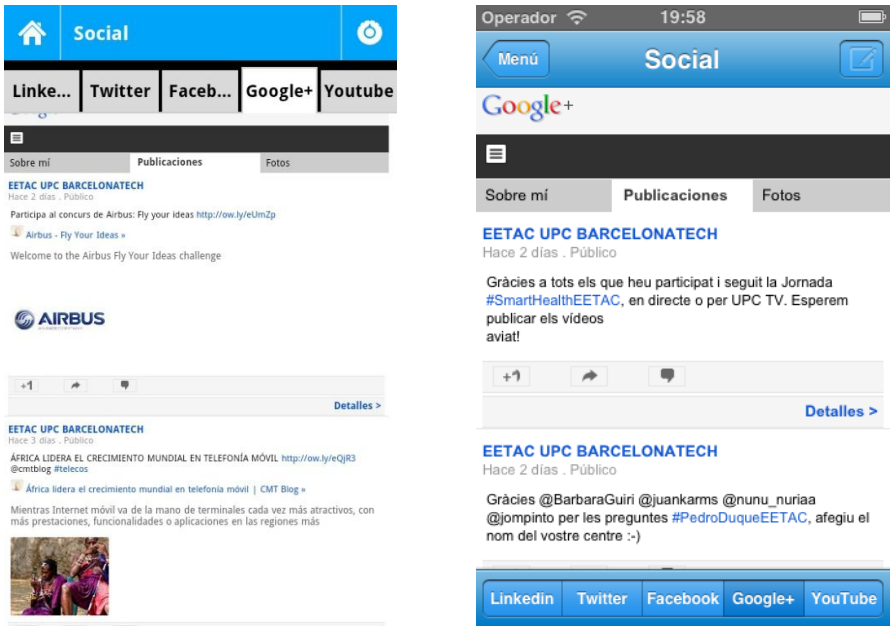

Fig. I.22 Xarxa social Google+, Android i iOS respectivament

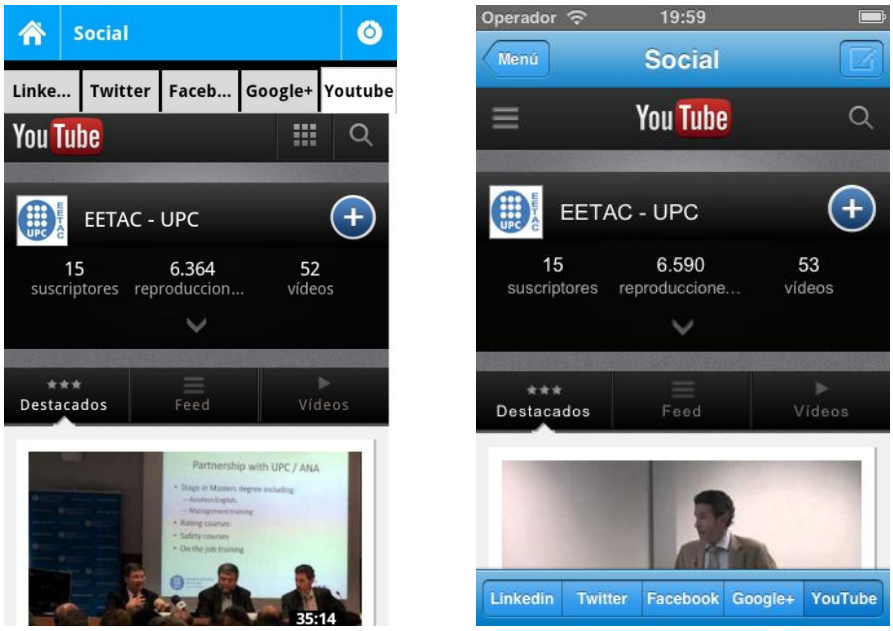

Fig. I.23 Xarxa de vídeos YouTube, Android i iOS respectivament

I.9. Mòdul de Dades

Fig. I.24 Llistat de dades i estadístiques, Android i iOS respectivament

Fig. I.25 Gràfica d'una estadística, Android i iOS respectivament

I.10. Mòdul d'Opcions

Fig. I.26 Menú de configuració, Android

Fig. I.27 Informació d'usuari procedent de l'EETAC, Android i iOS respectivament

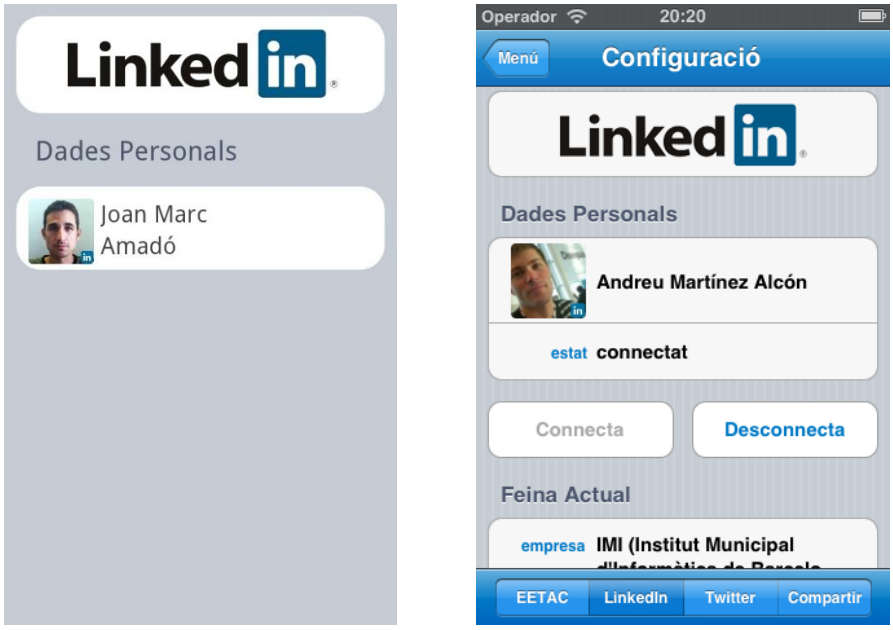

Fig. I.28 Informació d'usuari procedent de LinkedIn, Android i iOS respectivament

Fig. I.29 Informació d'usuari procedent de Twitter, Android i iOS respectivament

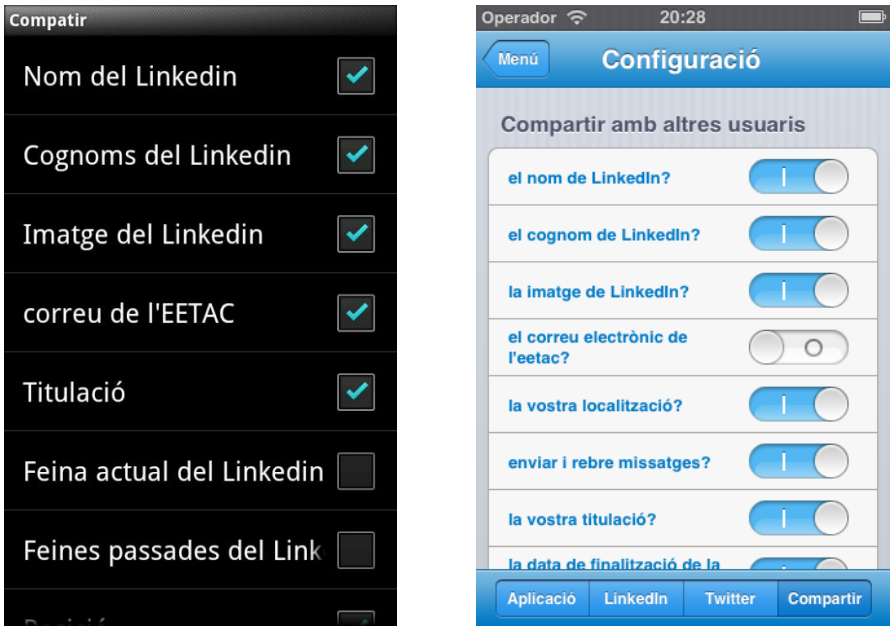

Fig. I.30 Vista per compartir la informació, Android i iOS respectivament

Annex II. ANDROID

II.1. Estructura d'una aplicació

Dins d'una aplicació d'Android hi ha quatre components principals: *Activity*, *Service*, *Content Provider* i *Broadcast Receiver*. Totes les aplicacions Android estan formades per algun d'aquest elements o una combinació de tots ells.

- ***Activity*:** és un component que proporciona una pantalla amb la qual els usuaris poden interactuar amb la finalitat de fer alguna cosa, com realitzar una trucada, fer una foto, enviar un correu electrònic, o veure un mapa.
- ***Service*:** Un Service (o Servei) és un tros de codi que s'executa en segon pla durant un temps indefinit sense necessitat d'una interfície gràfica. De manera que l'usuari pot seguir realitzant altres tasques. En cas de que hi hagi múltiples serveis a la vegada, se'ls pot indicar diferents prioritats segons les necessitats.
- ***Content Provider*:** En Android, les aplicacions poden guardar la seva informació en fitxers, bases de dades, etc. Però, en cas que es vulgui compartir la informació amb altres aplicacions, es necessita un *Content Provider*. Un *Content Provider* és una classe que implementa un conjunt estàndard de mètodes, que permeten a les altres aplicacions guardar i obtenir la informació que controla l'esmentat *Content Provider*.
- ***Broadcast Receiver*:** Aquest tipus de component, s'utilitza per rebre i reaccionar davant de certes notificacions broadcast. No tenen interfície gràfica i poden reacciona davant de d'esdeveniments com canvis de zona horària, trucades, nivell de bateria.

II.1.1. Android Manifest

En Android existeix un arxiu en format *XML* anomenat *AndroidManifest* que, tot i que no formi part del codi principal de l'aplicació, és necessari perquè funcioni correctament. Aquest fitxer es el fitxer de control que li diu al sistema que te que fer amb tots els components anteriorment mencionats i que conformen l'aplicació.

Com altres fitxers *XML* d'Android, el fitxer *AndroidManifest* inclou una declaració d'espai de noms. Aquesta declaració permet utilitzar una variable d'atributs estàndards d'Android. Tot Package presentat al usuari com aplicació d'alt nivell necessitarà incloure com a mínim un component <*activity*> que suporti l'acció *MAIN* i la categoria *LAUNCHER*, indicant que es tracte de l'activitat principal que s'inicia al executar l'aplicació.

És important destacar que les aplicacions d'Android no tenen permisos associats a no ser que es configurin prèviament en el *AndroidManifest*. D'aquesta manera s'eviten modificacions no desitjades.

II.1.2. Cicle de vida d'una aplicació

Cada aplicació Android córrer en el seu propi procés. Aquest procés és creat quant l'aplicació s'executa, i roman fins que l'aplicació deixa de treballar o el sistema necessita memòria per altres aplicacions. Una característica fonamental d'Android és que el cicle de vida d'una aplicació no esta controlat per la mateixa, sinó que ho determina el propi sistema a partir d'una combinació d'estats, d'aquesta manera, Android situa cada procés en una jerarquia "d'importància" basada en estats. Existeixen diferents processos d'acord a aquesta jerarquia:

- Un procés en primer pla és aquell que l'usuari requereix per lo que esta fent. Es considera en primer pla si s'està executant una Activitat en que l'usuari interactua amb ella, si s'està executant un *BroadcastReceiver*. O si s'està executant un servei.
- Un procés visible és aquell que conte una Activitat que és visible al usuari mitjançant la pantalla, però no en primer pla (esta pausada). Aquest procés solament s'eliminarà en cas que sigui necessari per mantenir en execució els processos en primer pla.
- Un procés de servei és aquell que conte un servei que ha sigut inicialitzat. No són directament visible al usuari i el sistema els mantindrà a no ser que no pugui mantenir els dos anteriors.
- Un procés en *background* és aquell que acull una activitat que no és actualment visible per l'usuari. Mentre aquests processos implementin bé el seu propi cicle de vida, el sistema pot eliminar-los per donar memòria a qualsevol dels 3 serveis anteriors.
- Un procés buit és aquell que no conté cap component actiu de cap aplicació. L'única raó per mantenir-lo és per millorar les seves inicialitzacions posteriors a mode de memòria cau.

Per comprendre millor el cicle de vida d'una aplicació d'Android, a la **figura II.1** es mostra el seu diagrama de flux, mencionant també els mètodes que es criden durant el transcurs del mateix.

II.1.3. Activity

És la principal classe d'Android, i cada pantalla que es mostra en l'aplicació és una classe que hereta de *Activity*. Així sent, podríem dir que una activitat equival a una pantalla. A la **figura II.1** es mostra el cicle de vida d'una activitat.

En el diagrama s'observen els mètodes que podem sobreescriure si volem fer una acció en concret, i quan s'executen. No cal ni obligatori implementar cap ells, l'únic que hem d'implementar sempre (almenys si volem que l'activitat mostri alguna cosa) és *onCreate*() on definim tot el vulguem fer abans que la pantalla es mostri. Per exemple en el cas que a una activitat es pugui tornar des de diverses activitats, i volem fer coses diferents depenent de quina vinguem, la forma idònia no seria amb *onResume* () ja que, si volem, en el

moment de cridar des d'una activitat a una altra podem deixar esperant el resultat de l'execució amb un mètode més eficient, que serà detallat quan s'expliqui la classe *Intent*.

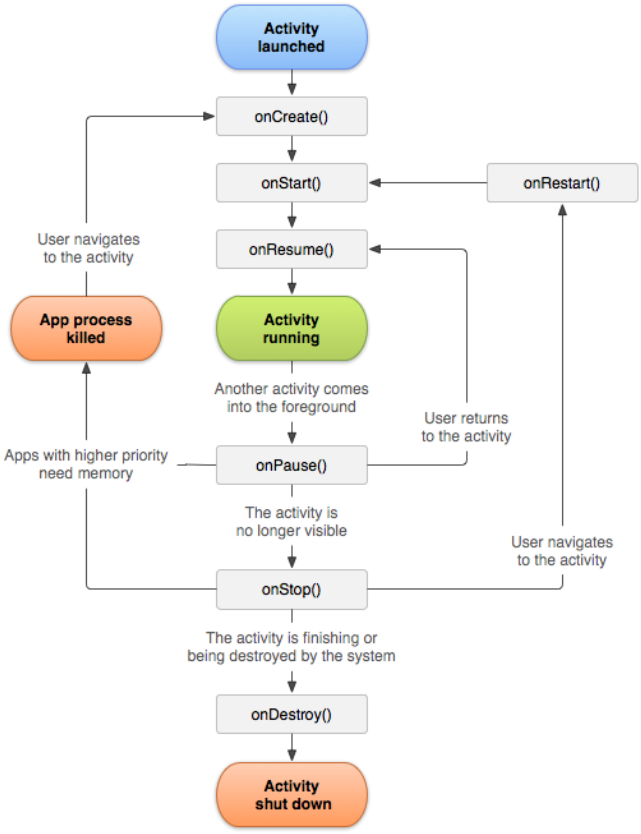

Fig. II.1. Cicle de vida d'una *Activity*

Com a curiositat, i cosa molt important a tenir en compte en el cas d'aquesta aplicació ja que gairebé totes les seves activitats llegeixen dades d'Internet, cal comentar que no hi ha mètode definit en el cicle de vida per detectar quan es canvia l'orientació de la pantalla. Si anem a llegir dades d'Internet, el normal és fer-ho dins el *onCreate()* ja que només s'executa una vegada i si passem a una altra activitat en tornar el *onCreate()* no s'executaria, de manera que no es tornen a llegir les dades. No obstant això, quan es canvia l'orientació, l'activitat és destruïda i es torna a executar el *onCreate()*. És a dir, si per descuit o per voluntat pròpia l'usuari mou el dispositiu i la pantalla gira, s'haurien de tornar a carregar totes les dades.

II.1.4. Intent

La millor manera de definir la classe Intent és com un nexe d'unió. El seu ús més freqüent és per passar d'un *Activity* a un altre, o per realitzar una acció en una activitat. Per entendreu millor direm que un Intent és una crida a una altra

aplicació, ja sigui nostra o del sistema operatiu Android. Dins d'aquesta crida, a més, podem afegir dades, ja siguin paràmetres de configuració o simple informació.

En aquesta línia de codi es mostra com passar de l'activitat que llista les gràfiques a la activitat que mostra informació més extensa de les mateixes. A més li enviem informació extra dient-li a quina direcció web s'ha de connectar per veure la gràfica desitjada.

Simplement s'ha de construir un objecte de classe Intent on els paràmetres siguin on estem i on volem anar:

```
Intent intent = new Intent(getApplicationContext(), GraphiquesWeb.class);
intent.putExtra("webs", webs[position]);
startActivity(intent);
```

Fig. II.2. Exemple d'execució d'un *Intent*

Al altre costat, la *Activity* ha de esta preparada per rebre les dades que li em passat al cridar-la, per fer això es fa mitjançant la següent línia:

```
intent.getStringExtra("webs");
```

Fig. II.3. Rebent informació extra

Com que en aquest cas el que s'envia és una variable del tipus *String*, el receptor ha de posar *getStringExtra*, per altres tipus de variables hi ha diferents comandes.

II.1.5. View

Tot objecte que volem que es mostri per pantalla heretar de la classe *View*. Podem definir les nostres pròpies classes que heretin de *View*, o utilitzar les que ens proporciona l'API d'Android. Els objectes *View* els podem organitzar i ordenar dins d'objectes de classe Layout. Amb un Layout, per exemple, podríem definir que els components s'agreguessin de manera horitzontal o vertical en una línia. També definir en un format de taula controlant les posicions en columnes i files. Entre els *layouts [93]* proporcionats per Android els següents són els més bàsics:

- ***LinearLayout:*** aquest *Layout* ordena als seus elements en una línia en sentit horitzontal o vertical, agregant-los un per un en l'ordre en què es van definint.
- ***FrameLayout:*** és el tipus més simple de *Layout* que permet afegir un sol element en un espai en blanc definit.
- ***TableLayout:*** aquest Layout afegeix els elements en columnes i files. Les files han de ser inserides una per una amb un *RowLayout*.
- ***RelativeLayout:*** permet afegir elements respecte a un element previ o els marges de la pantalla. Això significa que el posicionament s'administra segons la posició de l'element anterior.

Les interfícies es defineixen de forma jeràrquica, de manera que podem utilitzar tants elements de els descrits anteriorment com vulguem, combinant o introduint uns dins dels altres. Una altra forma d'organitzar els objectes de classe *View* és utilitzar els *ViewGroups*, els quals també poden contenir Layouts si es desitja. Aquests són els més comuns:

- ***Gallery:*** utilitzat per desplegar llistats d'imatges en un component amb *Scroll*.
- ***GridView:*** desplega una taula amb *Scroll* de m columnes i n files.
- ***ListView:*** desplega una llista amb *Scroll* d'una columna.
- ***ScrollView:*** una columna vertical d'elements amb *Scroll*. Dins de la jerarquia, el *ScrollView* només pot contenir un element, si volem que el scroll es realitzi sobre diversos *Views* aquests hauran d'estar dins d'un *ViewGroup* o *Layout* el qual serà el que estigui dins del *ScrollView*.

Un cop sabem organitzar els objectes de la classe View ja podem afegir els que desitgem. L'elaboració de la interfície serà explicada més endavant, ara simplement es farà una descripció dels objectes de la classe View més utilitzats:

- ***Button*****:** element que pinta un botó a la pantalla amb l'objectiu que l'usuari realitzi alguna interacció amb ell. Se li pot col·locar una imatge de fons, però per construir un botó que sigui una imatge existeix també una classe similar anomenada *ImageButton*.
- ***Checkbox*****:** caixa de text seleccionable que permet avaluar dos estats del element: seleccionat i no seleccionat.
- ***EditText*****:** element que permet editar text.
- ***TextView*****:** etiqueta de text no editable per l'usuari.
- ***RadioButton*****:** element seleccionable semblant al checkbox, però amb la diferència de que una vegada que s'ha seleccionat no torna al seu estat anterior si aquest està definit dins d'un *RadioGroup.*
- ***Spinner*****:** llistat d'elements ocult que només apareix quan es prem sobre ell. Quan està ocult només es pot veure l'element seleccionat de tot el llistat.

Amb aquestes tres classes, i els seus corresponents classes descendents, es podrien construir aplicacions bastant complexes. Podem resumir de la següent manera: una *Activity* és una pantalla a mostrar, tot el que vulguem que

aparegui a la pantalla ha de ser descendent de *View* o agrupacions d'aquests objectes com *Layouts* i *ViewGroups*. Finalment, els *Intent* ens permeten passar d'un *Activity* a un altre, o realitzar accions definides pel telèfon.

També hi ha la classe *Service*, de funcions similars a un *Activity* però executant-se en segon pla. És a dir, és com una activitat en background que podem estar executant i que no mostrarà res per pantalla durant la seva execució. A continuació, en lloc d'explicar classes concretes s'explicaran accions que s'han implementat per a aquesta aplicació, descrivint en cadascuna d'elles les classes de l'API necessàries per fer-ho.

II.1.6. Estructura de carpetes

II.1.6.1 SRC

En aquesta carpeta s'inclouen les classes amb el codi font de l'aplicació de la mateixa manera que en qualsevol altre projecte Java. Dins d'aquest directori el podem organitzar com vulguem, i amb totes les subcarpetes que siguin necessàries. En aquest cas dins de la carpeta SRC trobem que el projecte esta dividit en 4 subgrups:

- **com.socdeleetac:** Dintre d'aquesta carpeta trobem majoritàriament totes les classes que formen una Activity, que són visible en l'aplicació.
- **com.socdeleetac.clases:** Aquí és on he posat totes les classes que fan funcions no visible per l'usuari. Aquí hi ha classes com la Send(), que la seva funció és la de enviar dades al servidor, sempre que l'aplicació es connecta amb el servidor ho fa a traves d'aquesta classe.
- **com.socdeleetac.clases.Titulats:** aquesta carpeta conte totes les classes que defineixen a un titulat en l'aplicació. Són classes com les del anterior grup, no son visible per l'usuari, però s'encarreguen de definir-lo. Així doncs trobem classes com la de *EetacPerson()* que conte un seguit de variables i mètodes. Algun dels mètodes que conte aquesta classe és el de *getTitulacio*() que retorna una variable del tipus String [100] amb la titulació del usuari.

II.1.6.2 GEN

Aquesta carpeta és generada i controlada automàticament per l'entorn de desenvolupament, no s'ha de modificar ja que s'actualitza sola cada vegada que es faci algun canvi dins de la carpeta *res*. Serveix, per tant, com a interfície entre la carpeta res i el codi font contingut en *src*. Amb la qual cosa, cada vegada que vulgui accedir mitjançant codi a un element de la carpeta *res* (que serà explicada a continuació) s'ha de fer a través de la classe R.java.

II.1.6.3 RES

En aquesta carpeta s'inclou tot el que no sigui codi Java, però que vulguem que aparegui en l'aplicació. Per defecte estan creades les carpetes *drawable* (per col·locar imatges), *layout* (per crear arxius *XML* que defineixen interfícies) i *values* (per col·locar *Strings* i *arrays*). Però, com s'observa a la **figura II.4**, podem afegir més. En aquest cas tenim la carpeta *xml* on aquí hi ha un fitxer *XML* que defineix un el elements que es poden visualitzar en l'apartat de configuració. Cada vegada que introduïm un element nou en una d'aquestes carpetes, o modifiquem el contingut d'una cosa que ja fos, li assignarà un identificador a la carpeta *R.java* amb el nom que tingui l'arxiu si tot és correcte. En el cas dels arxius de la carpeta layout el nom ha d'estar escrit en caràcters en minúscules.

Finalment observem l'existència de l'arxiu *AndroidManifest.xml*, el qual té un paper molt important per al correcte funcionament de l'aplicació.

En ell s'han de definir tots els permisos dels que faci ús l'aplicació, ja que seran mostrats a l'usuari en el moment d'instal·lació perquè decideixi si la vol o no. Per exemple, si es vol fer us d'internet en el mòbil, com és el cas, però no definim en aquest arxiu el permís, l'aplicació llançarà una excepció i finalitzarà quan s'anés a produir aquesta acció. Per evitar aquest fet, escrivim la següent línia en l'arxiu

```
<uses-permission android:name="android.permission.INTERNET" />
```

Fig. II.4. Afegint permisos d'accés a internet al *AndroidManifest*

II.2. Maps

Una API (Application Programming Interface) consisteix en un conjunt de biblioteques de classes que permeten ser usades de forma fàcil i ràpida. D'aquesta manera es poden utilitzar i manejar molts elements en les aplicacions que ja han estat programats sense haver de començar des de zero.

El complement per les API de Google és una ampliació de l'entorn de desenvolupament del SDK d'Android que proporciona a les aplicacions un accés senzill als serveis a les dades de Google. La funció principal del complement és la biblioteca externa de Google Maps, que permet afegir funcions d'assignació potents a l'aplicació d'Android.

Per poder desenvolupar amb l'API de Google Maps és necessari una API key. Aquesta clau s'haurà d'inserir en el codi de l'aplicació perquè cada vegada que s'utilitzin els mapes del servidor de Google pugui veure que s'està registrat al servei. En el cas que no es disposi d'una clau o no sigui vàlida, l'aplicació segueix funcionant, però no es mostren els mapes.

II.2.1. Passos per aconseguir l'API Key de Google

Per aconseguir la clau de Google hem de crear una empremta digital de certificat anomenada MD5. Per aconseguir-la s'ha d'executar la consola de comandes de Windows. Un cop en la consola cal posicionar-nos en la carpeta bin de Java, on es troba el *keytools* (C:\Program Files\Java\jre7\bin), i s'introdueix la següent comanda:

```
keytool -list -alias androiddebugkey -keystore c:\Users\"nom_usuari"\.android\debug.keystore
-storepass android -keypass android
```

Fig. II.5. Comanda per genera empremta digital MD5

El "nom_usuari" es refereix el nom que una persona te en la sessió de Windows. Un cop fet això, apareixerà en pantalla la empremta digital. Tot seguit cal copiar-la i dirigir-se a la web següent:

```
https://developers.google.com/maps/documentation/android/maps-api-signup?hl=es-ES
```

Fig. II.6. Web de generació de la *API Key*

Un cop allà i després de haver-se llegit els termes de servei, cal acceptar-los i enganxar la empremta anteriorment obtinguda, premem la opció de "Generate API key" i si tot a anat correctament ens apareixerà per pantalla la clau necessària per fer funcionar els mapes en l'aplicació.

Ara que ja tenim la clau necessària, el que hem de fer a continuació és introduir-la al codi. En el *layout* que haguem creat per veure el mapa em de introduir la següent línia:

```
android:apiKey="09S4QgXek1vRP04QfpDH6A170RFcAdClEOXRgnw"
```

Fig. II.7. Afegint *ApiKey* al XML de la vista de Mapes

II.2.2. Objectes de Mapes

II.2.2.1 GeoPoint

La classe *GeoPoint* ve predefinida amb el paquet de mapes. Aquesta classe emmagatzema una posició en un mapa. Aquesta posició es representa mitjançant dos nombres de tipus double: latitud i longitud. Per passar d'aquests dos nombres a *GeoPoint* només cal convertir els nombres a micro graus elevant cada nombre a la sisena potència.

II.2.2.2 MapActivity i MapController

MapActivity és la classe que controla funcions bàsiques d'una *Activity* que mostra un mapa, mentre que *MapController* permet gestionar la panoràmica i el zoom del mapa.

II.2.2.3 MapView, MyLocationOverlay i Overlay

MapView és la vista que mostra el mapa, *MyLocationOverlay* permet dibuixar la posició actual de l'usuari i *Overlay* deixa posar objectes perquè apareguin al mapa.

II.2.3. Localització

Un altre aparta important relacionat amb els mapes és el de la localització. La localització es pot realitza principalment mitjançant dos mètodes el *GPS*, i la triangulació de antenes de telefonia mòbil. En el mètode de triangulació per antenes la precisió del sistema no és massa alta, la posició que indica al mapa pot estar en un radi de 500 a 5.000 metres de la teva posició real, depenent de la densitat d'antenes de telefonia mòbil amb què triangular el senyal del nostre propi mòbil. Tot i aquest fet, per la nostre aplicació no necessitem una precisió elevada, i el sistema de triangulació de antenes de telefonia és més ràpid i consumeix menys bateria. Però si l'usuari ho desitja pot activar el *GPS* del seu dispositiu i posicionar-se de forma més precisa. El primer que hem de fer si volem fer ús del sistemes de localització, és com sempre afegir permisos a l'aplicació en el Android Manifest amb les següents línies:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Fig. II.8. Afegint permisos de localització al AndroidManifest

La primer d'elles permet a l'aplicació accedeixi a un nivell alt de precisió de posicionament utilitzant el *GPS*. La segona és com la primera però amb un nivell més tosc de precisió.

II.3. Instal·lació GreenDroid

El mètode d'instal·lació d'aquesta llibreria difereix una mica de la resta, ja que no es tracta només de col·locar un arxiu en un carpeta determinada. Per instal·lar la llibreria, el primer que hem de fer és importar-la a l'Eclipse, per això, situem el cursor a la zona del *Package Explorer*, premem el botó dret > Import > Existing Projects into Workspace i a l'opció Select Root directory posem la ruta de la carpeta que conté la llibreria, en aquest cas GreenDroid.

El segon pas és afegir-la al nostre projecte, per a això hem de prémer el botó dret a la carpeta del nostre projecte a l'Eclipse > Properties, se'ns obrirà una finestra amb unes opcions al lateral dret, hem de seleccionar Android, baixar fins al final, i en Library > Add > GreenDroid > Ok.

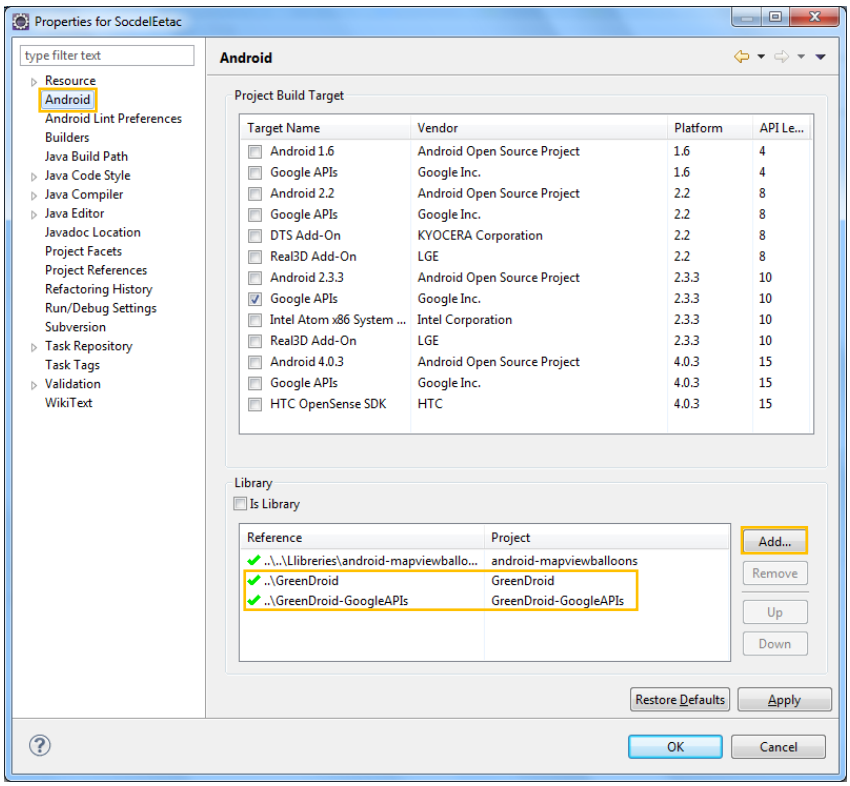

Fig. II.9. Afegir GreenDroid al programa

I per acabar, s'ha de repetir l'últim pas, afegint aquest cop la opció GreenDroid-GoogleAPIs, això és necessari, ja que, utilitzem la API de Google Maps en la nostre aplicació.

D'aquesta manera s'haurà afegit la llibreria al nostre projecte i estarà preparada per utilitzar-la.

II.4. Classe SendRecive

```
public class SendReceived {
        HttpPost httppost;
        HttpClient httpclient;
        HttpResponse response;

        List<WebsInputAmpl> llistaFinal;

        HttpGet httpget;
        final static String _WSCLIENTNAME = "RestManager";

        Gson gson;
        String serialized;
        StringEntity se;
        String serverIp = "147.83.XXX.XXX:XXXX";
        InputStream source;
        Reader reader;
        HttpEntity getResponseEntity;
        Messages messages;
        List<EetacPerson> persons;
        EetacPerson eetacpersons;
        StatisticsGroups[] groupStatistics;
        WebsGroups[] websGroups;


        public List<EetacPerson> sendImHere(ImHere imhere) {

                httppost = new HttpPost("http://" + serverIp
                                +
"/ImFromEetac/rest/ImFromEetac/getLocalitzationEetacPeople");
                gson = new Gson();
                serialized = gson.toJson(imhere);
                reader = setInfo();
                if (reader != null) {
                        try{
                persons = gson.fromJson(reader, new
                TypeToken<List<EetacPerson>>(){}.getType());
                        } catch (JsonIOException e) {
                                return null;
                        } catch (Exception e) {
                                return null;
                        }
                }
                return persons;
        }
        public Messages getRecived(RequestMessages rm) {
                httppost = new HttpPost("http://" + serverIp +
                "/ImFromEetac/rest/ImFromEetac/getReceivedMessages");
                gson = new Gson();
                serialized = gson.toJson(rm);
                reader = setInfo();
                if (reader != null) {
                        try{
                        messages = gson.fromJson(reader, Messages.class);
                        } catch (JsonIOException e) {
                                return null;
                        } catch (Exception e) {
                                return null;
                        }
                }
                return messages;
        }
```

```
	public Messages getSended(RequestMessages rm) {
		httppost = new HttpPost("http://" + serverIp +
		"/ImFromEetac/rest/ImFromEetac/getSendMessages");
		gson = new Gson();
		serialized = gson.toJson(rm);
		reader = setInfo();
		if (reader != null) {
			try{
			messages = gson.fromJson(reader, Messages.class);
			} catch (JsonIOException e) {
				return null;
			} catch (Exception e) {
				return null;
			}
		}
		return messages;

	}

	public List<EetacPerson> getChatPersonList(EetacPerson eetacPerson) {
		httppost = new HttpPost("http://" + serverIp +
		"/ImFromEetac/rest/ImFromEetac/getListEetacPeopleMessages");
		gson = new Gson();
		serialized = gson.toJson(eetacPerson);
		reader = setInfo();
		if (reader != null) {
			try{
			persons = gson.fromJson(reader, new
			TypeToken<List<EetacPerson>>(){}.getType());
			} catch (JsonIOException e) {
				return null;
			} catch (Exception e) {
				return null;
			}
		}
		return persons;
	}

	public EetacPerson sendEetacPerson(EetacPerson eetacperson) {

		httppost = new HttpPost("http://" + serverIp
			+"/ImFromEetac/rest/ImFromEetac/getLocalitzationEetacPeople");
		gson = new Gson();
		serialized = gson.toJson(eetacperson);
		reader = setInfo();
		if (reader != null) {
			try{
			eetacpersons = gson.fromJson(reader, EetacPerson.class);
			} catch (JsonIOException e) {
				return null;
			} catch (Exception e) {
				return null;
			}
		}

		return eetacpersons;
	}

	public EetacPerson postData(NewID newId) {

		httppost = new HttpPost("http://" + serverIp
				+ "/ImFromEetac/rest/ImFromEetac/getEetacPerson");
		gson = new Gson();
		serialized = gson.toJson(newId);
		reader = setInfo();
		if (reader != null) {
			try {
			eetacpersons = gson.fromJson(reader, EetacPerson.class);
			return eetacpersons;

			} catch (JsonIOException e) {
				return null;
			} catch (Exception e) {
```

```
                        return null;
                    }
                } else {
                    return null;
                }

        }

        public List<WebsInputAmpl> websReciver() {

                httpget = new HttpGet("http://" + serverIp
                        + "/ImFromEetac/rest/ImFromEetac/getWebsList");
                gson = new Gson();
                reader = getInfo();
                if (reader != null) {
                    try{
                    websGroups = gson.fromJson(reader, WebsGroups[].class);
                    WebsInputAmpl websInputAmpl;
                    List<WebsInput> websInput= new ArrayList<WebsInput>();
                    llistaFinal = new ArrayList<WebsInputAmpl>();

                    String groupTitle;
                    websInput = new ArrayList<WebsInput>();
                    for(WebsGroups temp:websGroups){
                        websInput = temp.getWebsGroup();
                        groupTitle = temp.getGroupTitle();
                        for(WebsInput temp2:websInput){
                            websInputAmpl = new WebsInputAmpl();

        websInputAmpl.setDescription(temp2.getDescription());
                            websInputAmpl.setUrlImage(temp2.getUrlImage());
                            websInputAmpl.setWebURL(temp2.getWebURL());

                                    websInputAmpl.setNavigationTitle
                                    (temp2.getNavigationTitle());
                            websInputAmpl.setGroupTitle(groupTitle);
                            llistaFinal.add(websInputAmpl);
                        }
                    }

                    } catch (JsonIOException e) {
                        return null;
                    } catch (Exception e) {
                        return null;
                    }
                }

                return llistaFinal;
        }

        public StatisticsGroups[] graphicReciver() {

                httpget = new HttpGet("http://" + serverIp +
                "/ImFromEetac/rest/ImFromEetac/getStatisticsList");
                gson = new Gson();
                reader = getInfo();
                if (reader != null) {

                    try{
                    groupStatistics = gson.fromJson(reader, StatisticsGroups[].class);

                    } catch (JsonIOException e) {
                        return null;
                    } catch (Exception e) {
                        return null;
                    }
                }

                return groupStatistics;
        }

        public Resposta sendAllows(EetacPerson eetacperson) {

                httppost = new HttpPost("http://" + serverIp
```

```
                        + "/ImFromEetac/rest/ImFromEetac/updateEetacPerson");
                gson = new Gson();
                serialized = gson.toJson(eetacperson);
                reader = setInfo();
                try{
                Resposta resposta = gson.fromJson(reader, Resposta.class);
                return resposta;
                } catch (JsonIOException e) {
                        return null;
                } catch (Exception e) {
                        return null;
                }
        }

        public Resposta sendMessage(Message message) {

                httppost = new HttpPost("http://" + serverIp
                                + "/ImFromEetac/rest/ImFromEetac/setMessage");
                gson = new Gson();
                serialized = gson.toJson(message);
                reader = setInfo();
                try{
                Resposta resposta = gson.fromJson(reader, Resposta.class);
                return resposta;
                } catch (JsonIOException e) {
                        return null;
                } catch (Exception e) {
                        return null;
                }
        }
```

Fig. II.10. Classe SendRecive, primer grup

```
        public Reader setInfo() {

                try {

                        HttpParams httpParameters = new BasicHttpParams();
                        int timeoutConnection = 4000;
                        HttpConnectionParams.setConnectionTimeout(httpParameters,
                                        timeoutConnection);
                        int timeoutSocket = 6000;
                        HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);

                        httpclient = new DefaultHttpClient(httpParameters);

                        se = new StringEntity(serialized, HTTP.UTF_8);
                        se.setContentType("application/json");
                        httppost.setEntity(se);
                        response = httpclient.execute(httppost);
                        getResponseEntity = response.getEntity();
                        source = getResponseEntity.getContent();
                        reader = new InputStreamReader(source);

                        return reader;

                } catch (ClientProtocolException e) {
                        e.printStackTrace();

                        return null;
                } catch (IOException e) {
                        e.printStackTrace();
                        return null;
                }

        }
```

```
        public Reader getInfo() {

                HttpParams httpParameters = new BasicHttpParams();
                int timeoutConnection = 4000;

        HttpConnectionParams.setConnectionTimeout(httpParameters,timeoutConnection);
                int timeoutSocket = 6000;
                HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);
                httpclient = new DefaultHttpClient(httpParameters);
                httpget.setHeader("content-type", "application/json");
                try {
                        HttpResponse result = httpclient.execute(httpget);
                        getResponseEntity = result.getEntity();
                        source = getResponseEntity.getContent();
                        reader = new InputStreamReader(source);
                        return reader;
                } catch (ClientProtocolException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                return null;

        }
}
```

Fig. II.11. Classe SendRecive, segon grup

Annex III. ACRÒNIMS I DEFINICIONS

III.1. Acrònims

[40] 3G: **3**th **G**eneration

[41] API: **A**pplication **P**rogramming **I**nterface

[42] CBL: **C**ampus del **B**aix **L**lobregat

[43] CSV: **C**omma-**S**epared **V**alues

[44] DB4O: **D**ata **B**ase **4** (for) **O**bjects

[45] EETAC: **E**scola d'**E**nginyers de **T**elecomunicació i **A**eroespacial de **C**astelldefels

[46] EPSC: **E**scola **P**olitècnica **S**uperior de **C**astelldefels

[47] EUPBL: **E**scola **U**niversitària **P**olitècnica del **B**aix **L**lobregat

[48] GPS: **G**lobal **P**ositioning **S**ystem

[49] HTTP: **H**ypertext **T**ransfer **P**rotocol

[50] HTTPS: **H**ypertext **T**ransfer **P**rotocol **S**ecure

[51] iOS: **i**Phone **O**peration **S**ystem

[52] IP: **I**nternet **P**rotocol

[53] JAXB: **J**ava **A**rchitecture for **X**ML **B**inding

[54] JSON: **J**ava**S**cript **O**bject **N**otation

[55] LTE: **L**ong **T**erm **E**volution

[56] MAST: **M**aster in **A**erospace **S**cience and **T**echnology.

[57] MASTEAM: **Ma**ster of **S**cience in **T**elecomunication **E**ngineering **a**nd **M**anagement

[58] REST: **RE**presentational **S**tate **T**ranfer

[59] SIA: **S**istema d'**I**nformació **A**cadèmica

[60] SOA: **S**ervice **O**riented **A**rchitecture

[61] UPC: **U**niversitat **P**olitecnica de **C**atalunya

[62] URL: **U**niform **R**esource **L**ocator

[63] W3C: **W**orld **W**ide **W**eb **C**onsortium

[64] XML: e**X**tensible **M**arkup **L**anguage

III.2. Definicions

[65] Activity: principal classe d'Android, utilitzada en cada pantalla que es mostra al usuari.

[66] Android: Sistema operatiu creat per Google per a mòbils i tablets.

[67] Apache Tomcat: Software per a la creació d'un servidor web

[68] API: Conjunt de biblioteques de classes que permeten ser usades de forma fàcil i ràpida.

[69] App Store: Servei creat per Apple, que permet als usuaris buscar i descarregar aplicacions.

[70] Back-end: part on es realitzen tasques alienes a l'usuari.

[71] Backoffice: part on es realitzen les tasques destinades a la gestió.

[72] Boolean: en programació, representa valors de lògica binaria.

[73] Bus SOA: arquitectura orientada a serveis, permet una millor gestió de la informació i serveis dels diferents components del bus.

[74] CSV: Tipus de document per representar dades en forma de taula.

[75] DB4O: Llibreria per al llenguatge Java per el emmagatzematge permanent de dades.

[76] Double: En programació, tipus de dada que representa l'aproximació d'un nombre real.

[77] Eclipse: És un entorn de desenvolupament integrat de codi obert multiplataforma.

[78] Geoposicionament: Traslladar una base de dades o un fitxer a un entorn espacial, generalment mapes digitalitzats.

[79] GET: mètode de HTTP per demanar informació.

[80] GreenDroid: Llibreria per Android per facilitar la col·locació d'una barra superior, entre altre coses.

[81] Gson: Llibreria per a Java utilitzada en Android, per la serialització i deserialització d'elements JSON.

[82] Hashtag: Cadena de caràcters formada per una o diverses paraules concatenades i precedides per un coixinet (#).

[83] Hipermèdia: Terme amb el qual es designa el conjunt de mètodes o procediments per escriure, dissenyar o compondre continguts que integrin suports com ara: text, imatge, vídeo, àudio, mapes.

[84] HTTP: Protocol usat en cada transacció de la World Wide Web.

[85] Int: En programació, tipus de dada que pot representar un subconjunt finit dels nombres enters.

[86] iOS: Sistema operatiu creat per Apple per a mòbils i tablets de Apple.

[87] iTunes: Reproductor de mitjans i botiga de continguts multimèdia desenvolupat per Apple.

[88] Jackson: Llibreria per a Java utilitzada en el servidor, per la serialització i deserialització d'elements JSON.

[89] Java: Llenguatge de programació orientat a objectes, utilitzat per a la creació de l'aplicació d'Android.

[90] JAXB: Eina que permet als desenvolupadors assignar classes de Java a representacions XML.

[91] Jersey: llibreria que implementa un servei REST amb format JSON.

[92] JSON: Format lleuger per a l'intercanvi de dades.

[93] Layout: recurs que defineix l'arquitectura de la interfície d'usuari en una Activity.

[94] List<Job>: En programació, seqüència de nodes on es guarden camps de dades arbitràries, en aquest es guarda un llistat d'objectes del tipus Job.

[95] Oauth: Protocol obert, que permet autorització segura d'un API de manera estàndard i simple per a aplicacions d'escriptori, mòbils, i web.

[96] Objective-C: Llenguatge de programació orientat a objectes, utilitzat per a la creació de l'aplicació de iPhone.

[97] POST: Mètode de HTTP per enviar o actualitzar informació.

[98] REST: Tècnica d'arquitectura de software per a sistemes hipermèdia distribuïts.

[99] Repiular: acte de re-publicar un tuit d'un altre usuari perquè tots els seus seguidors el llegeixin.

[100] String: En programació, una cadena de caràcters.

[101] Tuit: Missatge publicat via Twitter que conté 140 caràcters o menys.

[102] URL: Seqüència de caràcters, d'acord amb un format modèlic i estàndard, que s'usa per nomenar recursos a Internet per a la seva localització.

[103] Xcode: Entorn de desenvolupament integrat d'Apple.

[104] XML: Llenguatge de marques desenvolupat pel World Wide Web Consortium.